PROAGENTBENCH: EVALUATING LLM AGENTS FOR PROACTIVE ASSISTANCE WITH REAL-WORLD DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Proactive agents that anticipate user intentions without explicit prompts represent a significant evolution in human-AI interaction, promising to reduce cognitive load and streamline workflows. However, existing datasets suffer from two critical deficiencies: (1) reliance on LLM-synthesized data that fails to capture authentic human decision-making patterns, and (2) focus on isolated tasks rather than continuous workflows, missing the pre-assistance behavioral context essential for learning proactive intervention signals. To address these gaps, we introduce **ProAgentBench**, a rigorous benchmark for proactive agents in working scenarios. We study whether an agent can anticipate when a user will initiate an LLM interaction and what assistance the user will subsequently request, using only the workflow observations available before the interaction. Our contributions include: (1) a hierarchical task framework that decomposes proactive assistance into timing prediction and assist content generation; (2) a privacy-compliant dataset with 28,000+ events from 500+ hours of real user sessions, preserving bursty interaction patterns (burstiness B=0.787) absent in synthetic data; and (3) extensive experiments that evaluates LLM- and VLM-based baselines. Numerically, we showed that long-term memory and historical context significantly enhance prediction accuracy, while real-world training data substantially outperforms synthetic alternatives. We release our dataset and code at `https://anonymous.4open.science/r/ProAgentBench-6BC0`.

1 INTRODUCTION

Recent breakthroughs in large language models (LLMs) and embodied agent research have shifted the paradigm of human-AI interaction from reactive instruction following to proactive assistance (Lu et al., 2025; Sun et al., 2025; Yang et al., 2025b). As is illustrated in Figure 1, a proactive agent is defined as an AI system capable of perceiving environmental context, inferring user intentions without explicit prompts, and autonomously suggesting actions accordingly. Unlike reactive agents that rely on explicit user commands, proactive agents aim to bridge the gap between observable behavior and latent user needs. Prior HCI research has established that poorly timed interruptions impose significant cognitive costs on productivity (Mark et al., 2008), requiring users to expend additional mental resources for task resumption and context recovery (Iqbal & Horvitz, 2007). By anticipating user needs and providing timely, contextually appropriate assistance, proactive agents promise to reduce cognitive load and streamline task execution, enabling transformative productivity improvements in complex domains where human-AI collaboration is paramount.

Advancing proactive agents requires large-scale, high-quality datasets that capture authentic human-AI interaction patterns. However, existing datasets suffer from two critical deficiencies: (1) **Lack of real-world data**: Current datasets predominantly rely on LLM-synthesized interactions, which fail to capture the stochastic nature of human decision-making and the bursty temporal patterns inherent in real workflows (Mark et al., 2008). Agents trained on such data exhibit brittleness when facing real-world ambiguity, while scalable collection of authentic data is impeded by privacy concerns. (2) **Insufficient long-term data coverage**: Existing collections focus on isolated, short-duration tasks rather than continuous workflows, missing the *pre-assistance behavioral context*, the signals of what users were doing before seeking help. This context is essential for learning timing and content to proactively intervene.

To address these gaps, we present **ProAgentBench**, a rigorous benchmark designed to evaluate proactive agents in working scenarios. ProAgentBench benchmarks a precisely observable capability:

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

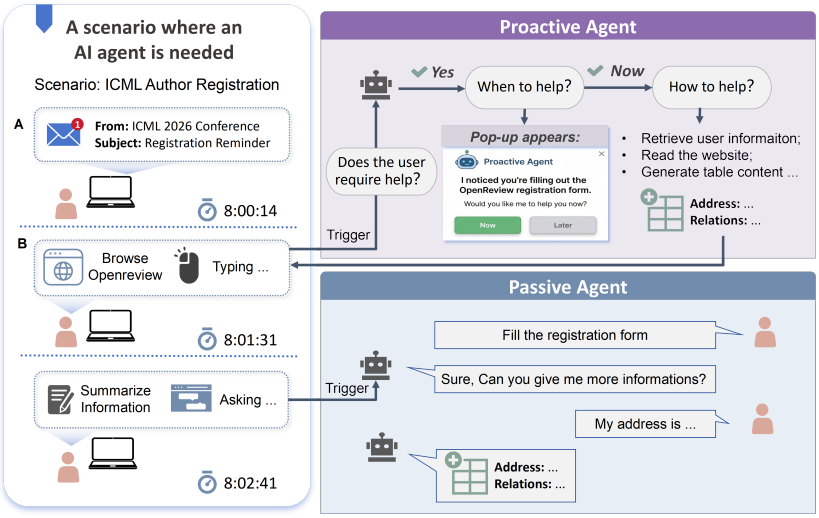


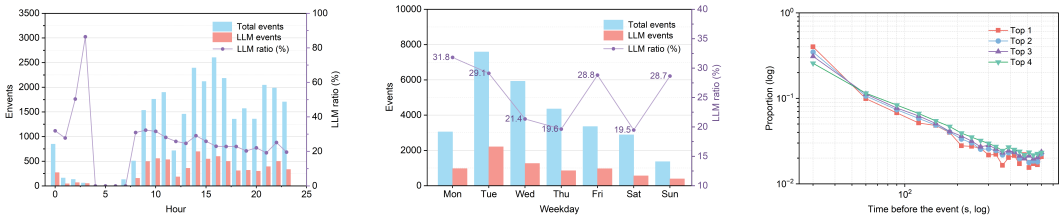
Figure 1: **Illustration of Proactive Agent Workflow.** The agent continuously monitors user screen activities and contextual signals. When assistance is needed, it proactively determines *when* to intervene and *how* to assist based on historical observations and user behavior patterns.

anticipating an imminent, user-initiated request for AI assistance and inferring its content from the preceding workflow. This behavioral endpoint is timestamped and reproducible, avoiding the recall bias and subjective. To tackle the **lack of real-world data**, we develop a privacy-compliant data collection pipeline that combines rule-based anonymization with human-in-the-loop review, enabling the safe collection of authentic user interactions at scale. Our dataset captures over 28,000 events from 500+ hours of continuous working sessions, preserving the bursty temporal patterns (burstiness $B = 0.787$) that synthetic data fundamentally lacks. To address **insufficient long-term coverage**, we record complete user work sessions rather than isolated tasks, explicitly capturing the *pre-assistance behavioral context*, namely, what users were doing in the minutes before seeking AI help, that is critical for learning proactive intervention signals. We then formalize a "**When + How**" hierarchical task framework that decomposes proactive assistance into two scientifically tractable problems: *When to Assist* (binary classification of optimal intervention timing) and *How to Assist* (generation of contextually appropriate content). This formalization enables systematic evaluation where each metric reflects real-world productivity impact: *precision* quantifies interruption cost (low precision causes alert fatigue (Schleyer & Thyvalikakath, 2012)), while *recall* measures need coverage (low recall fragments workflows (Adamczyk & Bailey, 2004)). In summary, our work makes three key contributions that directly address the identified gaps:

- We introduce **ProAgentBench**, the first rigorous benchmark for proactive agents providing a reusable paradigm for this emerging research area.
- We collect real-world human-AI interaction data with extensive user workflow logs, preserving authentic bursty interaction patterns and pre-assistance behavioral context, which are the critical signals preceding user needs.
- We conduct extensive experiments across diverse models and methods, revealing that both context and long-term memory significantly enhance prediction accuracy and real-world training data substantially outperforms synthetic data.

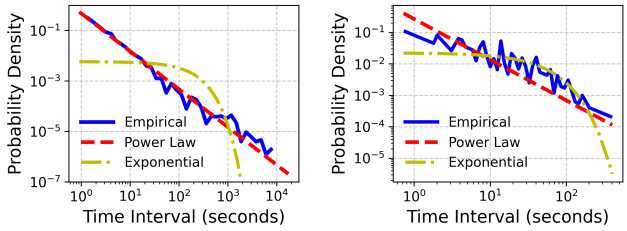
2 Related Work

Proactive Service Agents. Recent advances in LLMs have catalyzed significant progress in proactive agent research. Lu et al. (2025) pioneered data-driven proactive agent training with ProactiveBench, collecting 6,790 real-world events and achieving 66.47% F1-Score through reward modeling and fine-tuning. Yang et al. (2025b) introduced ContextAgent, which leverages multi-dimensional sensory perceptions from wearable devices to provide context-aware proactive assistance across 1,000 samples in daily life scenarios. In the mobile domain, Yang et al. (2025c) contributed FingerTip 20K, focusing on proactive task suggestions and personalized execution through long-term Android device interaction data. Liu et al. (2026) proposed ProactiveEval, a unified evaluation



(a) Weekday distribution. (b) Hourly (0–23) distribution. (c) Top-k similar screenshots.

Figure 2: **Temporal distributions and context relevance.** We report total events, LLM events, and the LLM ratio across (a) weekdays and (b) hours of day, and (c) distribution of time-to-event for Top-1/3/5/10 nearest screenshots (log-log). Similarity computed using `qwen2.5-vl-embedding`.



(a) Human Data (b) LLM Synthesized Data

Figure 3: Statistics of Human and LLM Synthesized Data

framework that decomposes proactive dialogue into target planning and dialogue guidance across 328 environments. However, these works remain limited in data scale, scenario coverage, and privacy protection mechanisms.

Screen Recording Datasets for Computer Use. The emergence of LLMs has driven demand for large-scale datasets capturing computer screen interactions. Pioneering efforts include Rico (Deka et al., 2017), which provided 72,000 mobile UI screenshots establishing foundations for data-driven interface analysis. For web environments, Deng et al. (2023) introduced Mind2Web with over 2,000 tasks across 137 websites, while Zhou et al. (2024) released WebArena with fully functional environments, revealing that even state-of-the-art models achieve only modest success rates. Desktop coverage expanded through AssistGUI (Gao et al., 2024) featuring professional software tasks, and OmniACT (Kapoor et al., 2025) with diverse desktop applications. Chen et al. (2025) contributed GUI-World with 12,379 video recordings highlighting temporal information importance, while Rawles et al. (2025) provided AndroidWorld with 116 parameterized mobile tasks. However, these datasets focus on task execution rather than proactive assistance, containing action sequences for predefined goals rather than organic user work patterns. Critically, they lack the *pre-interaction context* that captures what users were doing *before* seeking AI assistance, making it impossible to learn antecedent signals of user needs.

3 Problem Definition and Formulation

In this work, we define the proactive agent as an intelligent system that continuously monitors the user's current screen snapshots in real-time and proactively initiates contact upon detecting a need for service. Specifically, the proactive agent maintains a rich context of the user's historical information. Through this context, the agent achieves two goals: (1) modeling the user's inherent behavioral patterns, and (2) deriving the contextual background of the user's current snapshots. Based on these foundations, the proactive agent determines whether assistance is required and infers the user's intent, subsequently providing concrete, intent-aligned assistance and services. In this section, we first specify the inputs of proactive agent. After that, we provide a formal definition of the Proactive Agent and the data structures involved in our system. We then outline the hierarchical pipeline that decomposes the proactive assistance problem into two sub-tasks: timing prediction (*When to Assist*) and content generation (*How to Assist*).

Temporal Snapshot Sequence Inputs We define the input to a proactive assistance system as a temporal sequence of snapshots capturing user activities. At each time step i, the system captures a **snapshot** S_i consisting of multiple raw modalities: (1) **Screen Image** I_i: A screenshot capturing the

current visual state of the user's display, including application windows, UI elements, and on-screen content. (2) **Timestamp** τ_i: The precise time at which the snapshot was captured, enabling temporal analysis of user behavior patterns. (3) **Application Metadata** M_i: Contextual information including the active application name, window title, and application category. The historical observation sequence up to time t is thus defined as $O_{1:t} = \{S_1, S_2, \ldots, S_t\}$, where each $S_i = (I_i, \tau_i, M_i)$. In addition to real-time observations, the system has access to user meta-information U (e.g., role, preferences, and long-term memory derived from historical interactions). To avoid leakage, we require the proactive agent to make its prediction before the user opens the LLM. Therefore, the input does not include any information indicating that the user has opened the LLM. It only uses information available at prediction time, such as user meta-information U derived from historical interactions and the preceding context. Specifically, the user meta-information U comprises: (1) **Historical Information**: Records of the user's past interactions and long-term behavioral patterns, serving as a reference for contextual understanding. (2) **User Profile**: A structured model of the user, including attributes such as occupation, domain expertise, and specific preferences, which guides personalized assistance.

A Hierarchical Pipeline for Proactive Assistance Given the input sequence $O_{1:t}$ and user meta-information U, we design a two-stage pipeline that mirrors the natural decision process of an intelligent assistant. In the first stage (**When to Assist**), the agent continuously monitors user activities and determines the optimal moment to intervene, avoiding both premature interruptions that cause workflow disruption (Mark et al., 2008) and missed opportunities that force users to manually seek assistance. Only when the first stage predicts a positive trigger does the second stage (**How to Assist**) activate, generating contextually appropriate assistance content. This hierarchical design reflects the real-world constraint that unnecessary assistance queries (false positives in Stage 1) incur user interruption costs, while missed needs (false negatives) result in degraded user experience.

Task 1: When to Assist The interaction timing prediction is modelled as a binary classification problem that predicts whether proactive assistance is needed currently. Denoting the model as f_{when}, the prediction B_t is given by

$$B_t = f_{\text{when}}(U, O_{1:t}) \in \{0, 1\}$$

where $B_t = 1$ indicates that assistance is needed, and $B_t = 0$ indicates no intervention. The model f_{when} is implemented with LLMs. Our evaluation metrics are designed to directly reflect real-world productivity impact: (1) **Accuracy**: Measures overall system reliability, directly correlating with user trust and long-term adoption willingness. (2) **Precision**: Quantifies the rate of correct triggers among all interventions. Low precision leads to *alert fatigue*, excessive false alarms causing users to ignore or disable assistance features, ultimately degrading productivity (Schleyer & Thyvalikakath, 2012). (3) **Recall**: Captures coverage of actual user needs. Low recall means missed assistance opportunities, forcing users to manually seek help and fragmenting their workflow (Iqbal & Horvitz, 2007). (4) **F1 Score**: Trade-off between unnecessary interruptions (low precision) and missed opportunities (low recall), measuring effectiveness of proactive system.

Task 2: How to Assist When $B_t = 1$, the agent generates assistance content C_t:

$$C_t = f_{\text{how}}(U, O_{1:t}) \in \mathcal{V}$$

where $\mathcal{V}$ represents the natural language text space. The model f_{how} is implemented with LLMs. We evaluate the quality of generated assistance content using: (1) **Intention Accuracy**: Classification accuracy for coarse intention categories (see Appendix A.3). This metric reflects whether the agent correctly identifies the *type* of assistance needed (e.g., information retrieval vs. code generation), which determines the relevance of the response. (2) **Semantic Similarity**: Cosine similarity between predicted and real user query embeddings. This measures how well the generated content aligns with the user's actual query, directly impacting whether the assistance reduces or increases user effort.

4 Dataset Overview

4.1 Dataset Structure

To answer the two challenges: Impact of real-world data and long-term user context, we build up a dataset from real users with long-term user logs. We compare it with existing proactive assistance benchmarks in Table 1. Most of the existing benchmarks rely on synthetic training data or simulated environments and lack authentic long-term user context. For instance, Lu et al. (2025) uses synthetic training scenarios, Sun et al. (2025) employs simulated user feedback, and Yang et al. (2025b) relies on fabricated scenarios. Such reliance limits their ability to capture the natural dynamics of real-world

Table 1: Comparison with representative proactive-agent datasets/benchmarks. "Mixed" indicates that the resource is constructed by combining real-world signals with synthetic/simulated components. "LLM Queries" refers to timestamped records of user interactions with AI assistants, providing direct evidence of when and how users seek assistance.

Dataset	Real-World Data	Pre-Event Logs	Long-Term Context	LLM Queries	Original Info	#Events
Lu et al. (2025)	Mixed	✓	✗	✓	✗	6,790
Sun et al. (2025)	Mixed	✗	✗	✓	✗	6,563
Yang et al. (2025b)	Mixed	✓	✗	✓	✓	1,000
Yang et al. (2025c)	✓	✓	✓	✗	✓	21,437
Liu et al. (2026)	Mixed	✓	✓	✓	✗	328
ProAgentBench (Ours)	✓	✓	✓	✓	✓	**28,528**

workflows. In contrast, our dataset is derived entirely from continuous, real-world user activity logs, providing both pre-assistance behavioral traces and long-term context. This authentic, large-scale data enables robust evaluation of both *when* and *how* to assist within a unified, realistic framework.

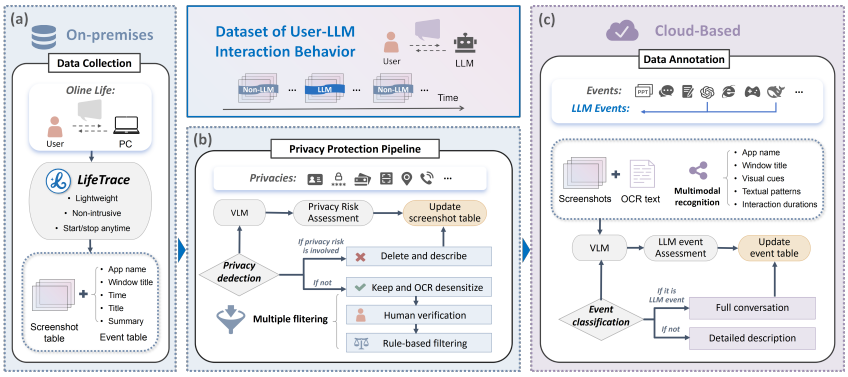


Figure 4: Data Collection Pipeline Overview. The figure illustrates the data collection process, including screenshot capture, metadata synchronization, privacy filtering, and storage workflow.

4.2 Dataset Statistics and Analysis

User Profile Statistics Our dataset comprises $N = 17$ student participants, spanning late undergraduate years and master's programs, and covers diverse academic backgrounds (e.g., computer science, electronic information, finance, biomedicine, energy, and translation). We collect 28,528 total events, among which 7,222 are LLM-related ($\approx 25.3\%$). To characterize usage purposes, we categorize LLM interactions by event semantics, including Information Retrieval (35.10%), Knowledge Q&A (20.42%), Data Analysis (9.17%), Code Programming (8.72%), and Content Generation (6.94%). From an application perspective, LLM-related events occur predominantly in web browsers (62.53%), and appear consistently in file management tools (11.34%), IDEs (9.25%), and office software (9.14%). Finally, at the platform level, identifiable providers are led by DeepSeek (23.62%) (DeepSeek-AI et al., 2024) and Gemini (18.69%) (Reid et al., 2024), alongside ChatGPT, Cursor, Doubao, and Kimi (Bai et al., 2025).

Temporal Usage Statistics To further characterize temporal usage patterns, we summarize weekday-level and hour-of-day distributions of total events, LLM events, and the LLM ratio; see Figures 2a and 2b. We also analyze pre-event context by retrieving the most semantically similar screenshots within a 10-minute window before each LLM event, as illustrated in Figure 2c. Specifically, we compute similarity between the LLM conversation-text embedding and screenshot image embeddings using Qwen2.5-VL-embedding (Wang et al., 2024), where top-k denotes the set of the k most similar screenshots. We observe the existence of a power-law relationship in the temporal distribution of relevant context. This finding underscores the critical importance of incorporating historical data, as key triggers for user needs are often buried in earlier interactions rather than being immediately adjacent to the current event.

Bursty Human-LLM Interaction. Let $\{t_i\}_{i=1}^{N}$ denote the time stamps of observed interactions, sorted such that $t_{i+1} \geq t_i$. We define the inter-event time (IET) as $\Delta t_i = t_{i+1} - t_i$, where $i = 1, \ldots, N-1$. To quantify whether the IETs exhibit a heavy tail, we fit a power law on the tail of $\{\Delta t_i\}$. Specifically, we assume $p(\Delta t) \propto (\Delta t)^{-\alpha}$, and estimate the exponent α using maximum likelihood. We compare the power law fit against an exponential alternative using a log-likelihood

Table 2: Performance comparison of prompt-based methods across different models. We report metrics for both the *When to Assist* task (Accuracy, Precision, Recall, F1 Score) and the *How to Assist* task (Intention Accuracy, Semantic Similarity). The best result in each column is **bolded**, and the second best is underlined.

Model	Method	When to Assist				How to Assist	
		Accuracy	Precision	Recall	F1 Score	Intent. Acc.	Sem. Sim.
Closed-Source Models							
GPT-4o-mini	Zero-shot	54.9%	52.7%	96.2%	68.1%	28.4%	0.280
	CoT	55.7%	55.6%	**99.5%**	**71.3%**	30.5%	0.298
	Self-Consistency	55.2%	52.8%	96.0%	68.2%	28.2%	0.280
Qwen3-Max	Zero-shot	59.3%	55.5%	93.4%	69.7%	36.3%	0.285
	CoT	59.8%	59.6%	72.5%	65.4%	**38.2%**	**0.305**
	Self-Consistency	59.5%	55.7%	93.5%	69.9%	36.2%	0.285
Deepseek-V3.2	Zero-shot	**64.4%**	60.8%	81.1%	69.5%	36.5%	0.276
	CoT	61.1%	**60.9%**	86.6%	**71.3%**	35.0%	0.287
	Self-Consistency	**64.4%**	60.8%	81.1%	69.6%	36.5%	0.276
Qwen3-VL-Plus	Zero-shot	53.0%	51.6%	97.0%	67.4%	37.1%	0.286
	CoT	53.5%	54.9%	61.3%	57.9%	34.4%	**0.305**
	Self-Consistency	53.1%	51.7%	97.0%	67.4%	36.7%	0.286
Open-Source Models							
Llama-3.1-8B-Instruct	Zero-shot	57.3%	54.7%	85.7%	66.7%	32.3%	0.275
	CoT	50.8%	50.4%	99.0%	66.8%	29.1%	0.294
	Self-Consistency	58.8%	56.7%	85.3%	68.1%	32.5%	0.274
Qwen3-VL-8B-Instruct	Zero-shot	51.7%	50.9%	94.4%	66.1%	35.3%	0.276
	CoT	41.0%	32.7%	17.1%	22.4%	34.1%	0.277
	Self-Consistency	52.9%	51.8%	93.6%	66.7%	35.7%	0.274

ratio test, where a positive log-likelihood ratio indicates that the power law provides a better fit, and a negative value favors the exponential model. We report the burstiness score B using IET (Goh & Barab'asi, 2008):

$$B = \frac{\sigma_{\Delta t} - \mu_{\Delta t}}{\sigma_{\Delta t} + \mu_{\Delta t}}, \tag{1}$$

where $\mu_{\Delta t}$ and $\sigma_{\Delta t}$ denote the sample mean and sample standard deviation of $\{\Delta t_i\}$, respectively. By construction, $B \in [-1, 1]$, with larger values indicating stronger temporal clustering and more bursty interaction patterns. In the human interaction records, the IET distribution is heavy-tailed and is well fitted by a power law with exponent $\alpha = 1.50$ (Fig. 3a). A log-likelihood ratio test strongly supports the power law over the exponential model (log-likelihood ratio $= 2951.48$, $p = 7.83 \times 10^{-100}$). The burstiness score is also high ($B = 0.787$), consistent with many short gaps and a few long gaps. For the synthetic data, we keep the same candidate time points as in the human records and let the LLM decide at each time point whether to interact. Under this setting, the IET distribution becomes closer to an exponential form on a log-log plot (Fig. 3b). In this case, the likelihood ratio test indicates that the exponential model fits better (log-likelihood ratio $= -59.36$, $p = 8.37 \times 10^{-7}$), and the burstiness score drops to $B = 0.166$. This suggests that even with realistic candidate time points, the LLM does not naturally reproduce the bursty timing in human behavior.

5 Data Collection, Privacy Protection, Automatic Annotation

We employ LifeTrace[1] to collect real-world computer usage data. To construct a high-quality, privacy-compliant dataset, we design a pipeline consisting of three main phases: data collection, privacy protection, and automatic annotation, as is illustrated in Figure 4.

Data Collection and Quality Control We collect user screen screenshots at 1Hz and synchronized application usage logs. Continuous user activities are automatically segmented into discrete events based on application switching. To ensure dataset quality, we implement a multi-layered filtering mechanism that excludes invalid events (e.g., extremely short duration or missing screenshots) and applies hash-based deduplication. Detailed setups and filtering criteria are provided in Appendix B.

Privacy Protection We prioritize user privacy through a rigorous three-stage process combining automated detection and human oversight. First, a local hosted VLM performs preliminary screening to identify sensitive visual and textual content. Second, we implement a human-in-the-loop mechanism where volunteers review and have final control over data retention. Finally, a rule-based filtering system acts as a safety net to catch remaining sensitive patterns. High-risk data is permanently deleted. The detailed privacy protocol is described in Appendix B.4.

[1] https://github.com/FreeU-group/LifeTrace

Automatic LLM Event Annotation To identify LLM interaction scenarios, we develop an event-level automatic annotation workflow. Our approach aggregates multi-modal context (image sequences, OCR text, and metadata) within an event window. We utilize Qwen3-VL-Plus (Yang et al., 2025a) to classify LLM platforms, interaction types, and extract conversation history after privacy protection. Specific prompt designs and annotation logic are detailed in Appendix B.5.

6 Experiments and Results

6.1 Experimental Setup

We utilize all interaction events occurring within the past 5 minutes as the historical context. This window captures the immediate workflow continuity while minimizing noise from stale activities. We evaluate a diverse set of state-of-the-art LLMs and Vision-Language Models (VLMs), including both closed-source (GPT-4o-mini (Hurst et al., 2024), Qwen3-VL-Plus, Qwen3-Max (Yang et al., 2025a), Deepseek-V3.2 (DeepSeek-AI et al., 2024)) and open-source (LLaMA3.1-8B-Instruct (Dubey et al., 2024), Qwen3-VL-8B-Instruct (Yang et al., 2025a)) models. For inferences, we adhere to the default hyperparameters provided by the respective model APIs or official repositories to ensure a fair and reproducible baseline comparison. Further, we implement a data splitting and evaluation protocol. First, we isolate each user's interaction history to prevent any cross-user information interference. Second, we employ time-based splits to mimic real-world deployment scenarios and avoid temporal data leakage. Finally, for the interaction timing prediction task, we carefully select non-assistance moments that are contextually similar to actual assistance triggers. This strategy filters out trivial negatives (such as periods of inactivity), forcing the model to distinguish between subtle differences in user needs and providing a more realistic benchmark for proactive assistance.

6.2 Base Results: Performance of Prompt-based Methods for Different Models

We first evaluate the effectiveness of state-of-the-art LLMs on proactive assistance using three prompt-based baselines we designed for this task: **Zero-shot**, **Chain-of-Thought (CoT)** (Wei et al., 2022), and **Self-Consistency** (Wang et al., 2022). While CoT and Self-Consistency are general prompting strategies, we adapt them with task-specific prompt designs tailored to the proactive assistance setting (see Appendix D for detailed prompt templates). Table 2 presents the comprehensive results across both tasks.

Zero-shot Performance. Among all evaluated models, Deepseek-V3.2 achieves the highest accuracy of 64.4% on the *When to Assist* task. Notably, closed-source models generally outperform their open-source counterparts. For the *How to Assist* task, Qwen3-VL-Plus achieves the best intention prediction accuracy of 37.1%. However, the semantic similarity scores remain relatively low across all models (ranging from 0.275 to 0.286), indicating that even state-of-the-art LLMs struggle to generate assistance content that closely matches user expectations.

Impact of Chain-of-Thought Prompting. We observe that CoT prompting yields mixed results depending on model capacity. For larger models, CoT improves *when to assist* performance. However, for smaller open-source models, CoT can be detrimental. This aligns with recent findings that CoT degrades performance on tasks involving implicit pattern recognition rather than explicit logical deduction Liu et al. (2025); Zheng et al. (2025). Our analysis reveals that CoT amplifies models' inherent behavioral tendencies: in Deepseek-V3.2 and LLaMA3.1-8B, CoT shifts decision boundaries toward aggressive triggering (higher Recall), while in Qwen3-VL-8B, it induces excessive conservatism (lower Recall). We further observe that CoT tends to overthink simple scenarios, imagining future problems rather than assessing what the user actually needs in the present, as illustrated in Figure 9. On the *How to Assist* task, CoT provides modest improvements in semantic similarity, indicating that structured reasoning helps models better articulate assistance content.

Self-Consistency Analysis. Self-Consistency sampling demonstrates stable but limited improvements over Zero-shot baselines. For instance, Llama-3.1-8B-Instruct improves from 57.3% to 58.8% accuracy, while Qwen3-VL-8B-Instruct improves from 51.7% to 52.9%. The F1 scores remain largely consistent across models. Notably, Self-Consistency does not significantly enhance intention prediction accuracy or semantic similarity, suggesting that the bottleneck lies in the models' fundamental understanding of user needs rather than output consistency.

Key Observations. Our experiments reveal several important findings: (1) The proactive assistance task remains challenging, with the both best accuracy on timing prediction and intention prediction remains low; (2) The gap between timing prediction accuracy and intention prediction accuracy suggests that *when to assist* is easier to determine than *how to assist*; (3) Advanced prompting techniques may harm performance; (4) The low semantic similarity scores indicate substantial room for improvement in generating contextually appropriate assistance.

6.3 Research Question 1: Impact of Historical Observation Sequence Length

To systematically evaluate the impact of historical information on proactive assistance, we conduct an ablation study on the historical observation sequence length $O_{1:t}$. This parameter controls the temporal range of user interaction logs provided to the model. We specifically investigate six distinct time settings: **10 seconds**, **30 seconds**, **1 minute**, **2 minutes**, **5 minutes**, and **10 minutes**. These settings allow us to analyze how the model's performance with different context ranging from immediate short-term history to more extended behavioral sequences.

We evaluate the performance using closed-source models. As summarized in Figure 7 (Appendix), both tasks benefit from longer historical context, though with different magnitudes. For the *when to assist* task, extending the context window yields gradual gains in F1 (Figure 7d), indicating that richer behavioral history helps distinguish assistance-needed moments from normal activities. For the *how to assist* task, intention accuracy improves as the window expands (Figure 7e). Intention accuracy shows diminishing returns beyond the 5-minute mark, with marginal gains between 5 and 10 minutes, so a 5-minute window balances behavioral coverage and cost.

This finding aligns well with the semantic relevance analysis presented in Figure 2c. While the majority of highly relevant events appear within a short time window immediately preceding the LLM interaction, the relevance distribution exhibits a pronounced long-tail effect. Specifically, although the most semantically similar events cluster within the first few minutes, a non-negligible portion of contextually important information spans beyond this immediate horizon. This long-tail characteristic explains why longer context windows are particularly beneficial for the *How to Assist* task: accurately inferring user intention often requires capturing sporadic but critical historical cues that may occur several minutes prior to the current interaction, even if they are temporally distant.

6.4 Research Question 2: Impact of Long-Term User Context

To evaluate the long-term user context in proactive assistance, we investigate the impact of incorporating long-term user behavior patterns. We introduce several **memory-based methods** that allow the agent to reference historical interaction data. Specifically, we implement and compare three distinct memory retrieval and organization strategies: (1) **Retrieval-Augmented Generation (RAG)** (Lewis et al., 2020), which retrieves via semantic similarity; (2) **Knowledge Graph (KG)**, which structures user habits into a relational graph; and (3) **Clustering**, inspired by the PersonaX approach (Shi et al., 2025), which categorizes user behaviors into distinct archetypes.

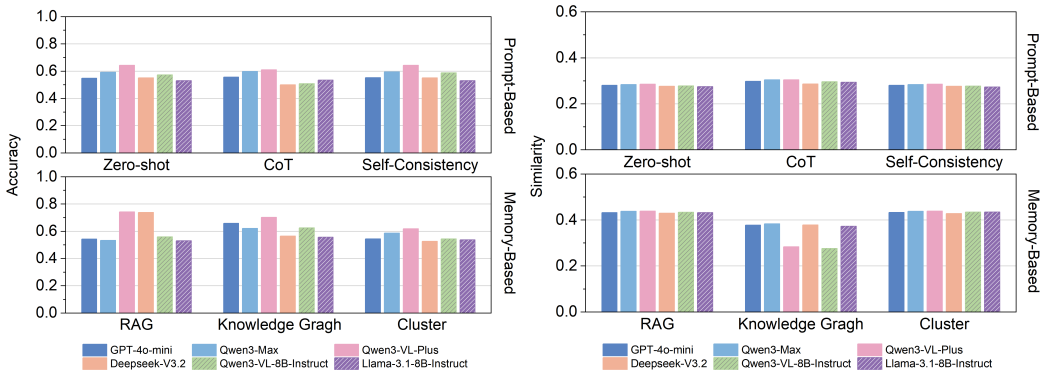
We observe that: (1) Incorporating long-term user behavior patterns via memory-based methods significantly enhances the effectiveness of personalized AI in proactive assistance, with Knowledge Graph (KG) emerging as the most optimal strategy. Among the three memory retrieval and organization approaches, KG achieves the most substantial performance improvement over the Zero-shot baseline, increasing overall Accuracy by 11.8% (from 0.537 to 0.601), Intention Accuracy by 26.9% (from 0.312 to 0.396), and F1 Score by 6.1% (from 0.675 to 0.716); (2) RAG demonstrates moderate effectiveness in leveraging historical interaction data, providing incremental improvements compared to the baseline without user behavior modeling. Specifically, RAG Memory-Based method improves Accuracy by 2.4% (reaching 0.550), Intention Accuracy by 6.3% (reaching 0.332), and maintains a stable F1 Score with a 0.8% increase (reaching 0.681), indicating its ability to reference relevant historical snippets effectively but with limited reasoning capability compared to KG.

6.5 Research Question 3: Impact of Real-World Data

A fundamental question in developing proactive assistance systems is whether real-world human data provides unique value compared to synthetic data generated by LLMs. We investigate whether task-specific fine-tuning on real-world data can substantially improve model performance, and how this compares to training on LLM-synthesized data.

We construct two training sets: (1) **Real-world data**: 741 instances sampled from our dataset, comprising diverse user profiles and authentic interaction patterns; (2) **Synthetic data**: An equivalent number of instances generated following Sun et al. (2025). **Fine-tuning Methods.** We employ two parameter-efficient fine-tuning strategies to adapt pre-trained models: (1) **Supervised Fine-Tuning (SFT)**: Learning rate of 2×10^{-5}, batch size of 16, for 3 epochs; (2) **Low-Rank Adaptation (LoRA)**: Fine-tuning with rank $r = 16$, learning rate of 2×10^{-4}, batch size of 16, for 3 epochs.

As shown in Table 3, both fine-tuning methods significantly enhance model performance compared to zero-shot baselines. Notably, for LLaMA-3.1-8B-Instruct, fine-tuning on real-world data leads to



(a) Performance on the *When to Assist* task.

(b) Performance on the *How to Assist* task.

Figure 5: **Performance comparison of different models and methods.** We apply both prompt-based methods (Zero-shot, CoT, Self-Consistency) and memory-based methods (RAG, Knowledge Graph, Cluster). (a) Accuracy on timing prediction. (b) Semantic similarity on content prediction.

Table 3: Impact of training data source on open-source models. We compare Zero-shot baseline with models fine-tuned on real-world vs. synthetic data. Abbreviations: Acc.=Accuracy, Int. Acc.=Intention Accuracy, FAR=False Alarm Rate, Sem. Sim.=Semantic Similarity. FAR is lower-is-better (↓); all other metrics are higher-is-better. Best results per model are **bolded**.

Method	Data	Acc.	F1	Int. Acc.	FAR ↓	Sem. Sim.
LLaMA-3.1-8B-Instruct						
Zero-shot	–	57.3%	66.7%	32.3%	42.3%	0.275
SFT	Synthetic	62.1%	70.2%	34.8%	35.6%	0.312
	Real-world	**74.0%**	**78.5%**	**42.1%**	**21.4%**	**0.385**
LoRA	Synthetic	60.5%	68.9%	33.6%	38.1%	0.298
	Real-world	71.2%	76.3%	40.5%	27.8%	0.372
Qwen3-VL-8B-Instruct						
Zero-shot	–	51.7%	66.1%	35.3%	47.1%	0.276
SFT	Synthetic	54.8%	67.8%	36.2%	41.2%	0.295
	Real-world	**63.5%**	**72.4%**	**41.8%**	**26.5%**	**0.358**
LoRA	Synthetic	53.2%	67.1%	35.9%	43.4%	0.288
	Real-world	61.8%	71.2%	40.2%	31.6%	0.345

substantial improvements, boosting Accuracy from 57.3% to 74.0% (+16.7%). More importantly, models trained on real-world data consistently outperform those trained on synthetic data across all metrics. For instance, LLaMA with SFT on real-world data achieves 74.0% Accuracy compared to 62.1% with synthetic data (+11.9%), and Intention Accuracy improves from 34.8% to 42.1% (+7.3%). This performance gap demonstrates that authentic human interaction patterns contain valuable signals that synthetic data cannot fully replicate. This finding underscores the critical importance of collecting and utilizing real-world human data for developing effective proactive assistance systems.

7 Conclusion

In this paper, we present **ProAgentBench**, a rigorous benchmark designed to evaluate proactive agents within continuous real-world workflows. Addressing the limitations of synthetic and isolated datasets, we construct a privacy-compliant dataset capturing over 28,000 events from 500+ hours of authentic user activities, preserving critical pre-assistance behavioral patterns. We propose a hierarchical framework to evaluate proactive capabilities in timing prediction and content generation. Our experiments reveal that real-world training data and long-term memory integration are pivotal for agent performance. Still, our cohort consists of 17 student participants in Chinese-language desktop environments, so the behavioral statistics we report, including the burstiness level $B = 0.787$, characterize this cohort rather than a general working population. Our positive labels also record when a user chose to initiate an LLM interaction, which is an observable proxy for, but not identical to, the moment at which assistance would have been most useful. Going forward, we plan to extend the benchmark with deployment-oriented studies of usability and intrusiveness. We hope these efforts will help ProAgentBench serve as a practical foundation for context-aware proactive assistants in real workflows.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks with required disclosure. (i) *Generating synthetic data.* The synthetic training set used in the real-versus-synthetic comparison (Section 6.5) was generated by Qwen3-VL-14B following the protocol of Sun et al. (2025). The synthetic interaction timings in Section 4.2 were produced by prompting an LLM to decide, at each candidate time point taken from the human records, whether to initiate an interaction. (ii) *Cleaning and annotating the dataset.* Qwen3-VL-Plus performed first-pass privacy screening of screenshots (Section B.4). It also identified LLM-interaction events and their platforms, generated event summaries, and extracted conversation text (Section B.5). (iii) *Qualitative and thematic data analysis.* The intention taxonomy was constructed through an LLM-assisted bottom-up procedure, and intent labels were assigned with LLM assistance (Section A.3). In addition, generative models served as components of our experimental pipeline. An LLM generated the natural-language personas for the Cluster baseline (Appendix C.3), and Qwen-Max acted as an automatic judge in the supplementary semantic-alignment evaluation (Appendix G.3). All LLMs and VLMs evaluated in Section 6 and Appendices F–H are objects of study rather than research assistants. We have not used generative AI tools to develop theoretical models or conceptual frameworks, formulate or prove mathematical claims, propose hypotheses, design research methodology or experiments, implement methods, translate material, or interpret results. Additionally, we used generative AI tools to edit the manuscript for grammar, clarity, and style, and to check consistency between reported numbers, tables, and text.

We have reviewed all AI-assisted work. Automatic event annotations were validated by manual inspection of 100 randomly sampled events (Section B). VLM privacy screening was followed by participant review, in which participants retained final authority over retention, and then by deterministic rule-based filtering. Synthetic data was used only as a controlled comparison condition. All AI-suggested text edits were checked by the authors against the original technical content to ensure that no new claims, citations, or results were introduced. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All participants in this study were voluntary contributors who were fully informed about the data collection process. Prior to participation, each individual was clearly briefed on: (1) the types of data collected, including screenshots and application metadata; (2) the research purpose and potential academic publication; (3) comprehensive privacy protection measures; and (4) the unconditional right to withdraw at any time with complete data deletion guaranteed within 7 days.

To protect participant privacy, we implemented a rigorous three-stage pipeline: real-time filtering of sensitive applications (e.g., banking, medical), VLM-based automatic detection of personally identifiable information (names, phone numbers, emails, passwords), and mandatory participant review where individuals retained final authority over all retention decisions. Participants could mark any screenshot for deletion without providing justification.

We acknowledge that screen-level behavioral data carries inherent surveillance risks if misused. To mitigate these concerns, we enforce strict access controls: raw screenshots are restricted to approved researchers under signed data use agreements, while public releases contain only de-identified data and aggregated statistics. All usage is governed by a research-only license that explicitly prohibits re-identification attempts, commercial applications, and any form of user monitoring or profiling. We believe these comprehensive safeguards ensure that the scientific benefits of ProAgentBench substantially outweigh potential risks.

REFERENCES

Piotr D Adamczyk and Brian P Bailey. If not now, when? the effects of interruption at different moments within task execution. In *International Conference on Human Factors in Computing Systems*, pp. 271–278. ACM, 2004. doi: 10.1145/985692.985727.

Kimi Team Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ning-Xin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, et al. Kimi K2: Open agentic intelligence. *arXiv*, 2025.

Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, et al. GUI-World: A video benchmark and dataset for multimodal GUI-oriented understanding. In *arXiv*, 2025.

DeepSeek-AI, A. Liu, B. Feng, Bing Xue, Bing-Li Wang, Bochao Wu, Cheng-Da Lu, Cheng-Gang Zhao, C. Deng, Chenyu Zhang, et al. DeepSeek-V3 technical report. *arXiv.org*, 2024.

Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *ACM Symposium on User Interface Software and Technology*, pp. 845–854, 2017. doi: 10.1145/3126594.3126651.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems 36*, 36:28091–28114, 2023. doi: 10.52202/075280-1220.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, A. Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv*, 2024.

Difei Gao, Lei Ji, Ze-Chen Bai, Ming-Yu Ouyang, Peiran Li, Donzxing Mao, Qinchen Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, et al. AssistGUI: Task-oriented PC graphical user interface automation. In *Computer Vision and Pattern Recognition*, pp. 13289–13298. IEEE, 2024. doi: 10.1109/CVPR52733.2024.01262.

Kwang-Il Goh and Albert-L'aszl'o Barab'asi. Burstiness and memory in complex systems. *EPL (Europhysics Letters)*, 81(4):48002, 2008. doi: 10.1209/0295-5075/81/48002.

Yizheng Huang and Jimmy Xiangji Huang. A survey on retrieval-augmented text generation for large language models. *ACM Computing Surveys*, 58(12):1–38, 2024. doi: 10.1145/3805774.

OpenAI Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card. *arXiv*, 2024.

Shamsi T Iqbal and Eric Horvitz. Disruption and recovery of computing tasks: field study, analysis, and directions. In *International Conference on Human Factors in Computing Systems*, pp. 677–686. ACM, 2007. doi: 10.1145/1240624.1240730.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem AlShikh, and Ruslan Salakhutdinov. OmniACT: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *Lecture Notes in Computer Science*, pp. 161–178. Springer Nature Switzerland, 2025. doi: 10.1007/978-3-031-73113-6_10.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, F. Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, M. Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020.

Ryan Liu, Jiayi Geng, Addison J. Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L. Griffiths. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse. In *International Conference on Machine Learning*, 2025. doi: 10.48550/arXiv.2410.21333. URL https://openreview.net/forum?id=J3gzdbYZxS.

Tianjian Liu, Fanqi Wan, Jiajian Guo, and Xiaojun Quan. ProactiveEval: A unified evaluation framework for proactive dialogue agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 41068–41100. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1906.

Yaxi Lu, Shen-Zhi Yang, Cheng Qian, Gui-Fang Chen, Qin-Yu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, et al. Proactive Agent: Shifting LLM agents from reactive responses to active assistance. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.12361.

Gloria Mark, Daniela Gudith, and Ulrich Klocke. The cost of interrupted work: more speed and stress. In *International Conference on Human Factors in Computing Systems*, pp. 107–110. ACM, ACM, 2008. doi: 10.1145/1357054.1357072.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, J. Waltz, G. Lau, Marybeth Fair, Alice Li, William E. Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2405.14573.

Machel Reid, N. Savinov, Denis Teplyashin, Dmitry Lepikhin, T. Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv.org*, 2024.

Chris Richardson, Yao Zhang, Kellen Gillespie, Sudipta Kar, Arshdeep Singh, Zeynab Raeesy, Omar Zia Khan, and Abhinav Sethy. Integrating summarization and retrieval for enhanced personalization via large language models. In *arXiv.org*, 2023. doi: 10.48550/arXiv.2310.20081.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. LaMP: When large language models meet personalization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7370–7392. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.399.

Titus Schleyer and Thankam P. Thyvalikakath. Alert fatigue. *The Journal of the American Dental Association*, 143(4):332–333, 2012. doi: 10.14219/jada.archive.2012.0166.

Yunxiao Shi, Wujiang Xu, Zhang Zeqi, Xing Zi, Qiang Wu, and Min Xu. PersonaX: A recommendation agent-oriented user modeling framework for long behavior sequence. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5764–5787. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.300.

Weiwei Sun, Xuhui Zhou, Weihua Du, Xingyao Wang, Sean Welleck, Graham Neubig, Maarten Sap, and Yiming Yang. Training proactive and personalized LLM agents. *arXiv.org*, 2025. doi: 10.48550/arXiv.2511.02208.

Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. Democratizing large language models via personalized parameter-efficient fine-tuning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6476–6491. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.372.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Ke-Yang Chen, Xue-Jing Liu, Jia-Lin Wang, Wenbin Ge, et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv.org*, 2024. doi: 10.48550/arXiv.2409.12191.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Neural Information Processing Systems*, volume 35, pp. 24824–24837. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022. doi: 10.52202/068431-1800.

An Yang, An-Feng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bo-Wen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv*, 2025a.

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. In *Advances in Neural Information Processing Systems 38*, pp. 185637–185671. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025b. doi: 10.52202/085713-5582.

Qinglong Yang, Haoming Li, Haotian Zhao, Xiaokai Yan, Jingtao Ding, Fengli Xu, and Yong Li. FingerTip 20K: A benchmark for proactive and personalized mobile LLM agents. *arXiv.org*, 2025c. doi: 10.48550/arXiv.2507.21071.

Tianshi Zheng, Yixiang Chen, Chengxi Li, Chunyang Li, Qing Zong, Haochen Shi, Baixuan Xu, Yangqiu Song, Ginny Wong, and Simon See. The curse of CoT: On the limitations of chain-of-thought in in-context learning. *Trans. Mach. Learn. Res.*, 2025. ISSN 2835-8856. doi: 10.48550/arXiv.2504.05081. URL https://openreview.net/forum?id=7SIrvcYNYj.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. doi: 10.48550/arXiv.2307.13854.

A Data Release & Usage

The dataset will be released in tiered access levels to balance research utility with privacy protection. Raw screenshots are restricted to approved researchers under strict data use agreements. The public release includes de-identified screenshots, derived features, event-level summaries, and evaluation protocols.

All usage is governed by a research-only license that explicitly prohibits re-identification attempts and commercial applications. We provide privacy-minimizing training and evaluation guidelines, along with reproducible scripts and baseline implementations to facilitate adoption by the research community.

A.1 Data fields

To support reproducible analyses and downstream modeling, we release the dataset in fully parseable, structured formats, consisting of (i) a SQLite database that stores the core logs and (ii) external annotation/curation files (CSV/JSON/JSONL) that provide traceable labeling and filtering decisions. The database is organized around two primary tables, including `screenshots` and `events`, which are linked through a consistent key (`event_id`), while the external files are keyed by the event identifier (e.g., `user` + `event_id`) to enable deterministic alignment with database entries.

Screenshot record fields. The `screenshots` table provides the record-level information required to locate a screenshot, align it on the timeline, and assess its integrity. It includes the screenshot file name (`file_path`) and creation time (`created_at`), along with aligned foreground context (`app_name`, `window_title`). In addition, we store file-level attributes such as content hash (`file_hash`), file size (`file_size`), and image dimensions (`width`, `height`). These attributes enable systematic reporting of data quality (e.g., missing files, duplicates, and abnormal file properties) and provide deterministic signals for integrity checks. Because some `file_path` values preserve capture-side absolute paths, reproducible usage typically resolves screenshots by filename and matches them to the local screenshot directory.

Event fields. The `events` table provides the event-level representation. It includes an event identifier (`id`), temporal boundaries (`start_time`, `end_time`), and event-level context (`app_name`, `window_title`). Events also contain an LLM-related flag (`is_llm_event`) and textual descriptors (e.g., `event_summary`, `detailed_description`, and optionally model-generated titles/summaries). In addition, events store a structured conversation field (`conversation`) as a JSON string, which captures observable interaction cues such as extracted user queries and model responses (e.g., `user_queries`, `llm_responses`, and `full_conversation`). These event fields support platform inference, semantic categorization, and topic representation based on visible input content.

External annotations and curation artifacts. In addition to database-native fields, we provide external files that make labeling and dataset curation explicit and auditable. These artifacts include (1) recheck/exclusion lists that specify which candidate events should be removed and why (e.g., verified non-LLM cases or unusable entries), (2) intent annotation outputs (e.g., `user_intention` with confidence and optional rationales) stored in JSONL/JSON for deterministic replay, and (3) when available, platform tags (e.g., `llm_platform`) from verification pipelines that improve the stability of platform attribution beyond app/window heuristics. All external artifacts are indexed by event keys and can be joined back to the database unambiguously, ensuring that the final analysis set and its labels are fully traceable and reproducible.

Overall, the combination of database fields and external annotation/curation files provides a complete, parseable interface to the dataset: the former captures the core screenshot and event logs, while the latter records reproducible labeling and filtering decisions required to construct the analysis-ready subset used in this work.

A.2 Data organization and file structure

The dataset is organized with participant (user) as the top-level key. For each participant, we provide both the screenshot files and the structured metadata required to parse and align the logs.

At the file-system level, each participant directory contains a `screenshots/` folder that stores the screenshots sampled at approximately 1 Hz. Screenshot filenames encode date and time information, which supports convenient retrieval at the day/hour granularity when needed. Some participant folders also include additional pipeline-generated subdirectories (e.g., privacy-processing artifacts), which may contain copies of screenshots or related intermediate outputs.

At the metadata level, each participant directory includes a SQLite database file named `lifetrace_privacy_processed.db`. The database is centered around two core tables, `screenshots` and `events`: the `screenshots` table stores screenshot-level records (timestamps, foreground app, window title, and file attributes), and the `events` table stores event-level records (event boundaries, context, and semantic fields). These tables are linked via `screenshots.event_id` and `events.id`, allowing each event to be mapped to its associated screenshot sequence and each screenshot to be traced back to its parent event.

In addition, some semantic annotations (e.g., event intent labels) are stored as separate JSON/JSONL files. These files are indexed by event keys (participant/user + `event_id`), enabling straightforward alignment with the event records in the SQLite databases.

A.3 Intention Categories

To characterize usage intent, we categorize LLM interactions into multiple scenario types based on event semantics. Table 4 presents the complete distribution of user intentions. Overall, information-seeking needs dominate: information lookup and knowledge Q&A together account for over 55% of all LLM events. Meanwhile, productive and analytical tasks also represent a substantial share, including data analysis, coding/programming, and content generation. The remaining categories form a long-tail distribution.

Table 4: Distribution of user intentions across LLM interaction events.

Intention Category	**Percentage**	**Count**
Information Lookup	35.10%	2,535
Knowledge Q&A	20.42%	1,475
Data Analysis	9.17%	662
Coding/Programming	8.72%	630
Content Generation	6.94%	501
Advice/Consultation	5.23%	378
Summarization	4.74%	342
Document Editing	3.28%	237
Comparison/Evaluation	2.17%	157
Translation	1.27%	92
Creative Design	1.16%	84
Format Conversion	0.79%	57
Mathematical Calculation	0.40%	29
Uncategorized	0.55%	40
Functional Testing	0.03%	2
Error Correction	0.01%	1
Total	**100.00%**	**7,222**

How the intention taxonomy is constructed. The **full** categorical schema in Table 4 is obtained by an LLM-assisted, bottom-up procedure rather than a fixed hand-written list. Concretely, we (i) randomly sample LLM interaction events from the collected logs; (ii) prompt an LLM to read each event's observable context (applications, window titles, OCR text, and conversation snippets) and to propose recurring *user intents* in free form; (iii) cluster semantically equivalent intent phrases and merge them into candidate categories; (iv) iteratively refine the taxonomy by splitting overly broad clusters and merging overlapping ones until the label set stabilizes across passes; and (v) resolve ambiguous cases (e.g., a query plausibly spanning two intents) by taking the user's *primary* intent as indicated by the conversational thread. This process yields the multi-class intention space used for *How to Assist*; for evaluation, models first decide whether assistance is needed, and intention

accuracy is computed only on positive events where a user query is available. Semantic similarity is likewise computed only on positive samples, since negatives have no reference utterance.

B Data Collection, Privacy Protection, and Automatic Annotation

We employ LifeTrace[2] to collect volunteers' computer usage behavior data. Through a multi-stage privacy protection process and an event-based data annotation workflow, a high-quality dataset containing Large Language Model (LLM) interaction scenarios is constructed. The entire pipeline consists of three main phases: data collection, privacy protection, and automatic annotation, ensuring the authenticity, privacy security, and annotation accuracy of the data, as is illustrated in Figure 4.

B.1 Participants and Setup

We recruited a diverse group of participants spanning various age ranges and LLM usage frequencies, covering multiple professional scenarios. All participants were required to use devices running Windows 10+ or macOS 12+ to ensure compatibility with our data collection tools. The data collection was conducted over a month, during which participants had full control over the process. We implemented strict consent and withdrawal protocols, allowing participants to pause or exit the study at any time and request the deletion of their data, ensuring ethical compliance and user privacy.

B.2 Data Collection and Instrumentation

We use LifeTrace application to collect real-world computer usage data, which monitors user activities with the user's informed consent. LifeTrace collects two types of data: (1) User screen screenshots captured at a rate of 1Hz; (2) Application usage logs, recording timestamps, application names, and window titles. Based on application switching and temporal continuity, LifeTrace automatically segments continuous user activities into discrete events. Each event represents a complete interaction period of the user in a specific application and window environment.

B.3 Quality Filtering

We implement multiple filtering mechanisms to improve dataset quality. For quality filtering, the following criteria are applied: (1) Events with a duration of < 3 seconds are excluded, as they typically lack meaningful LLM interactions; (2) Events with an abnormally long duration (> 1 hour) are reviewed to rule out potential system anomalies; (3) Each event must be associated with at least one valid screenshot; (4) For LLM events, we prioritize retaining events with ≥ 3 screenshots to ensure complete conversation context; (5) We verify conversation records of LLM events, ensuring the conversation records is not empty.

The system integrates fault-tolerance mechanisms, including automatic API retries, JSON parsing error recovery, and default annotations when VLM fails. SHA-256 file hashing is used for screenshot deduplication to eliminate redundant data. This comprehensive quality control process yields a high-quality dataset with an annotation success rate of 97.6% (verified through manual inspection of 100 randomly sampled events).

B.4 Privacy Protection

Prior to annotation, a three-stage privacy protection process integrating automatic detection, manual verification, and rule-based filtering is implemented to ensure comprehensive privacy safeguards.

Phase 1: VLM-based Preliminary Judgment Qwen3-VL-Plus (Yang et al., 2025a) is applied to perform multimodal privacy detection on all screenshots as the first line of defense. The model analyzes visual content and OCR-extracted text to identify sensitive information, including names, phone numbers, email addresses, ID card numbers, bank card numbers, passwords, and facial images. For each screenshot, the VLM generates: (1) Privacy risk level (safe/moderate/high-risk); (2) Type of

[2]Project available at `https://github.com/FreeU-group/LifeTrace`

detected privacy information; (3) Recommended action (retain/blur/delete); (4) Scene description for potential replacement. This automated phase provides a high-recall preliminary screening to capture possible sensitive content.

Phase 2: Volunteer Correction To address the limitations of automatic detection, volunteers review screenshots marked as safe/moderate and recommended for retention by the VLM, and make a final decision for each image: retain, blur, or delete. This human-in-the-loop approach ensures that context-sensitive information missed by the VLM can be identified, and participants retain control over their own privacy boundaries. We provide volunteers with clear data review guidelines and examples to ensure their fully informed consent regarding data upload.

Phase 3: Rule-based Filtering After manual verification, a rule-based system conducts final validation to capture edge cases and enforce consistency. This phase applies deterministic rules, including: (1) Pattern matching for common privacy identifiers (regular expressions for phone numbers, email formats, and ID card numbers); (2) File metadata checks (e.g., screenshots with window titles containing specific keywords are flagged); (3) Consistency verification (e.g., if multiple screenshots in the same event are deleted, adjacent screenshots will be re-evaluated); (4) OCR text cleaning using predefined replacement patterns. Screenshots that pass all three phases are migrated to a new database with additional privacy-related metadata fields, while high-risk screenshots are permanently deleted and replaced with scene descriptions. Critically, the original OCR text is deleted to prevent privacy leakage.

B.5 Automatic LLM Event Annotation

We developed an event-level automatic annotation process to identify LLM usage scenarios. Unlike traditional screenshot-level classification, our method operates at the event level to leverage temporal context across multiple screenshots.

For each event, we first sample up to 6 screenshots (the first 3 and the last 3) to balance computational cost and information retention. Deleted screenshots are replaced with scene descriptions. We extract the first 500 characters from the OCR results of each screenshot and organize them in chronological order. These multi-modal inputs (images, OCR text, and event metadata including application name, window title, and duration) are fed into Qwen3-VL-Plus via a carefully designed prompt. Our prompt explicitly instructs the model to: (1) Determine whether the event represents an LLM usage scenario; (2) If positive, identify the LLM platform (ChatGPT/Claude/Cursor, etc.) and interaction type (text conversation/code generation/image generation/multi-modal); (3) Generate a concise event summary ($\leq$ 20 words); (4) Extract the complete conversation, including all user queries and LLM responses. The model output is constrained to JSON format for structured data extraction. The VLM identifies LLM usage through multiple features: (a) Visual cues, including chat interface layouts, brand logos (ChatGPT icon, Claude logo), and UI components (message bubbles, send buttons); (b) Text patterns in OCR results, such as alternating questions and answers, platform identifiers ("ChatGPT says:", "Claude:"), and generation markers (code blocks, mathematical formulas); (c) Metadata signals, including application names (cursor.exe, chrome.exe), window titles containing LLM platform names, and typical interaction durations ($>$ 30 seconds).

C Memory-Based Methods implementation Details

C.1 Retrieval-Augmented Memory

While the Knowledge Graph captures aggregated semantic priors, we implement a Retrieval-Augmented Generation (RAG) module to provide LLMs with user-specific historical context at inference time. This approach retrieves concrete historical events similar to the current context and injects them verbatim into the prompt as external memory. This setting follows standard RAG-based memory augmentation paradigms Huang & Huang (2024); Lewis et al. (2020), adapted here with strict temporal constraints.

Memory Construction. For each user u, we define their episodic memory $\mathcal{M}_u$ derived from their specific subset of the training data $\mathcal{D}_u = \{(a_i, w_i, u_i, h_i, y_i, t_i) \in \mathcal{D} \mid u_i = u\}$, where t_i denotes the

timestamp. We serialize each interaction record into a structured textual document d_i via a template $T(a_i, w_i, y_i)$, encompassing the active application a_i, window title w_i, and the ground-truth intent y_i. We then employ a fixed embedding model $\phi(\cdot)$ to map each document to a dense vector space:

$$\mathbf{v}_i = \phi(d_i), \quad \forall i \in \mathcal{D}_u \tag{2}$$

The resulting memory store $\mathcal{M}_u = \{(\mathbf{v}_i, d_i)\}_{i=1}^{|\mathcal{D}_u|}$ acts as a key-value index, constructed exclusively from training data to ensure strict user isolation.

Temporal-Constrained Retrieval. Given a query context $x = (a, w, u, h, t)$, we generate a query embedding $\mathbf{q} = \phi(T(a, w, \emptyset))$. To retrieve relevant context without violating causality, we enforce a strict temporal constraint ensuring only past events are accessible. The retrieval set $\mathcal{R}$ consists of the top-5 neighbors based on cosine similarity:

$$\mathcal{R} = \underset{(\mathbf{v}_i, d_i) \in \mathcal{M}_u}{\arg \; top{-}5} \left(\frac{\mathbf{q} \cdot \mathbf{v}_i}{\|\mathbf{q}\| \|\mathbf{v}_i\|} \right) \quad \text{s.t.} \quad t_i < t \tag{3}$$

This mechanism effectively filters out future information, simulating a realistic setting where the agent only has access to the user's history up to the present moment.

Prompt Augmentation. Each retrieved memory item is formatted as a *Memory Block* containing its application name, window title, and concise semantic descriptions. The retrieved memory blocks are concatenated to form the memory context $\mathcal{C}_{\text{RAG}}$, which is injected into the prompt alongside the task description:

$$\tilde{x} = x_{\text{context}} \oplus [\texttt{MEMORY} : \mathcal{C}_{\text{RAG}}] \oplus x_{\text{task}} \tag{4}$$

$\mathcal{C}_{\text{RAG}}$ presents the LLM with full narrative examples (e.g., *"In a similar context with VSCode, the user previously searched for generic syntax help"*). This allows the model to leverage few-shot in-context learning to refine its intent understanding based on precedent.

Complexity. Memory construction incurs $O(N)$ computational cost for a single pass of offline embedding. During inference, retrieval operates in $O(\log |\mathcal{D}_u|)$ time using Approximate Nearest Neighbor indexing, with end-to-end retrieval latency below 10 ms per query in practice. The method introduces no additional trainable parameters and incurs negligible cost relative to standard LLM inference.

C.2 Knowledge Graph Memory Augmentation

To capture user behavioral patterns across applications, we construct a lightweight knowledge graph (KG) from historical interaction data and use it to augment inference-time prompts with contextual priors. This memory-augmented approach follows recent work on personalized LLMs Salemi et al. (2024); Tan et al. (2024), where user-specific information is retrieved and prepended to prompts without model fine-tuning.

Graph Construction. Given a training set $\mathcal{D} = \{(a_i, w_i, u_i, h_i, y_i)\}_{i=1}^{N}$, where a_i denotes the active application (e.g., Chrome, VSCode, Word), w_i the window title, u_i the user identifier, $h_i = [a_i^{(1)}, \ldots, a_i^{(k)}]$ the recent application history, and $y_i = (b_i, c_i)$ the ground-truth labels for help-needed (binary) and intention (categorical), we construct a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with four node types: App (application software), Keyword (window title tokens), Intent (intention categories), and User.

For each application $a \in \mathcal{A}$, we compute empirical priors:

$$P_{\text{help}}(a) = \frac{|\{i : a_i = a \wedge b_i = \texttt{True}\}|}{|\{i : a_i = a\}|} \tag{5}$$

$$P_{\text{intent}}(c \mid a) = \frac{|\{i : a_i = a \wedge c_i = c \wedge b_i = \texttt{True}\}|}{|\{i : a_i = a \wedge b_i = \texttt{True}\}|} \tag{6}$$

Additionally, we extract keywords from window titles using tokenization and stopword filtering, then compute keyword-conditioned intent probabilities:

$$P_{\text{intent}}(c \mid k) = \frac{|\{i : k \in \text{keywords}(w_i) \wedge c_i = c\}|}{|\{i : k \in \text{keywords}(w_i)\}|} \tag{7}$$

where keywords$(\cdot)$ extracts the top-5 informative tokens from each window title.

We also track application transition patterns from the history sequence, adding directed TRANSITION edges between consecutive applications with frequency-based weights.

Context Retrieval. At inference, given a test sample (a, w, u, h), we query $\mathcal{G}$ to retrieve a context tuple:

$$\mathcal{C}(a, w, h) = \Big(P_{\text{help}}(a),\ \Pi_a,\ \Pi_w\Big) \tag{8}$$

where $\Pi_a = \{(c, P_{\text{intent}}(c \mid a)) : c \in \mathcal{I}_a\}$ denotes app-based intent priors, and $\Pi_w = \{(c, \bar{P}_{\text{intent}}(c \mid w))\}$ aggregates keyword-based priors by averaging over extracted keywords:

$$\bar{P}_{\text{intent}}(c \mid w) = \frac{1}{|K_w|} \sum_{k \in K_w} P_{\text{intent}}(c \mid k) \tag{9}$$

We retain only intents exceeding a frequency threshold $\tau = 0.05$ for app-based priors and $\tau = 0.1$ for keyword-based priors to reduce noise.

Prompt Augmentation. The retrieved context $\mathcal{C}$ is serialized into natural language and inserted before the task instruction in the prompt:

$$\tilde{x} = x_{\text{context}} \oplus [\texttt{MEMORY} : \mathcal{C}] \oplus x_{\text{task}} \tag{10}$$

The memory section presents distributional hints in interpretable form (e.g., "*Based on historical patterns for this application: Code Programming 45%, Knowledge Q&A 30%*"). This provides the model with empirical priors as soft guidance without constraining its predictions, allowing it to override historical patterns when current context suggests deviation Richardson et al. (2023).

Complexity. Graph construction requires a single pass over $\mathcal{D}$ with $O(N)$ time and $O(|\mathcal{A}| \cdot |\mathcal{I}| + |\mathcal{K}| \cdot |\mathcal{I}|)$ space, where $|\mathcal{K}|$ is the keyword vocabulary size. Inference-time retrieval operates in $O(|K_w|)$ for keyword lookup, adding negligible overhead (<1ms per query), making the approach suitable for real-time proactive assistance.

C.3 CLUSTER-BASED PERSONA MEMORY

While the Knowledge Graph captures statistical priors and RAG retrieves raw episodes, we implement a Cluster-Based Persona memory that summarizes a user's historical behaviors into a compact set of natural language personas. This approach first groups historical interactions into coherent behavior clusters and then uses a large language model to generate high-level textual descriptions for each cluster. The resulting descriptions serve as user-level behavioral personas and are injected into the prompt at inference time. Both the clustering and sampling procedures strictly follow the PersonaX protocol Shi et al. (2025).

Hierarchical Behavior Clustering. For each user u, we construct the persona set exclusively from the training history $\mathcal{D}_u$. To strictly prevent data leakage, all events occurring in the evaluation period are removed prior to construction. Each remaining event e_i is serialized into text (by concatenating its application name, window title, event summary and detailed description) before being embedded into a dense vector v_i using a fixed embedding model. We then perform hierarchical clustering over the event embeddings, following the PersonaX protocol without modification. Events are clustered separately for LLM-related and non-LLM-related activities, with a distance threshold of 0.7 and a maximum of 15 clusters per category.

Prototypical-Diverse Sampling. To summarize each cluster C_j within a limited token budget, we select a representative subset $\mathcal{S}_j \subset C_j$. We adopt the same greedy sampling strategy as PersonaX, which balances *prototypicality* and *diversity* within each cluster. Prototypicality favors events closer to the cluster centroid, while diversity encourages coverage of heterogeneous behaviors.
All sampling hyperparameters are set to the values reported in PersonaX, with a fixed sampling ratio of 0.6 and a trade-off weight $\alpha = 1.06$. Let μ_j be the centroid of cluster C_j. The scoring function for a candidate subset $\mathcal{S}_j$ is defined as:

$$\mathcal{J}(\mathcal{S}_j) = w_p \sum_{e \in \mathcal{S}_j} \frac{1}{1 + \|\mathbf{v}_e - \mu_j\|} + w_d \cdot \frac{2}{|\mathcal{S}_j|} \sum_{e_a, e_b \in \mathcal{S}_j} \|\mathbf{v}_a - \mathbf{v}_b\| \tag{11}$$

where $w_p = \alpha^{-10}$ and $w_d = 1 - w_p$. This mechanism ensures the selected events capture the cluster's core intent while covering heterogeneous behavioral patterns.

Persona Generation. For each cluster, we prompt a large language model to generate a single textual persona p_j that summarizes the user's behavioral patterns represented by the cluster. The prompt instructs the model to abstract specific actions into habitual preferences (e.g., *"User frequently consults API docs while coding"*) without revealing sensitive information. Each persona is constrained to $100 \sim 120$ tokens. This yields a persona bank $\mathcal{P}_u = \{p_1, \ldots, p_m\}$ for user u.

Persona Retrieval. At inference time, given a test context $x = (a, w, u, h)$, we retrieve the most relevant behavioral priors. We compute the cosine similarity between the query embedding $\phi(x)$ and each persona in $\mathcal{P}_u$. The top-k ($k = 5$) personas are retrieved to form the context set $\mathcal{P}^*$.

Prompt Augmentation. The retrieved personas are serialized and injected into the system prompt as explicitly labeled priors:

$$\tilde{x} = x_{\text{context}} \oplus [\texttt{PERSONA} : \mathcal{P}^*] \oplus x_{\text{task}} \tag{12}$$

The prompt structure and decoding strategy remain consistent with other baselines to ensure fair comparison.

Complexity. Persona construction requires a single embedding pass over historical events followed by hierarchical clustering, resulting in $O(N^2)$ (where N is the history length) worst-case time per user but with small N in practice. Persona retrieval at inference time scales linearly with the number of personas and introduces negligible overhead. This method introduces no additional trainable parameters and operates entirely at inference time.

D VLM PROMPTS

We design a structured prompting framework to process multimodal user activity data for proactive assistance prediction. Our approach supports three inference strategies: zero-shot, chain-of-thought (CoT) Wei et al. (2022), and self-consistency Wang et al. (2022), each with tailored prompt templates.

D.1 EVENT DETECTION PROMPT TEMPLATES

The system prompt establishes the model's role as a screen activity monitor:

> *"You are an intelligent assistant responsible for monitoring user screen activity. Based on the user's current screen state and recent behavior, determine whether the user needs help from an AI assistant."*

The user prompt follows a hierarchical structure with four components:

Recent Activity Context. A summary of user activities from the preceding 5-minute window, providing temporal context for behavioral pattern recognition.

Current State. Structured metadata including application name, window title, timestamp, and a brief screen summary (truncated to 200 characters for efficiency).

Screen Content. For vision-enabled models (e.g., Qwen2.5-VL Wang et al. (2024)), we pass the screenshot directly with the marker "[See attached image]". For text-only models (e.g., Llama-3.1-8B-Instruct Dubey et al. (2024)), we provide OCR-extracted text from the current screen.

Task Instruction. The query section varies by inference method. For zero-shot prompting, we request direct binary prediction with intention classification. For CoT, we decompose the task into a four-step reasoning process. For self-consistency, we use the zero-shot format with multiple sampling and majority voting.

D.2 Sequence Analysis and Intention Classification

We define 16 intention categories covering common assistance scenarios (e.g., knowledge Q&A, code programming, content creation, information retrieval). The model selects from this predefined taxonomy when predicting user intention, enabling consistent evaluation across methods.

For CoT prompting Wei et al. (2022), we explicitly structure the reasoning chain into four steps:

1. Describe what the user is currently doing
2. Analyze potential problems or needs the user may encounter
3. Judge whether AI assistance is needed (yes/no)
4. Classify the user's intention category from the predefined set

This decomposition encourages the model to ground its prediction in observable screen evidence before committing to a classification, following the principle that intermediate reasoning steps improve complex task performance Wei et al. (2022).

For self-consistency Wang et al. (2022), we sample multiple reasoning paths using temperature-based decoding and aggregate predictions via majority voting. This approach leverages the intuition that complex reasoning tasks typically admit multiple valid reasoning paths leading to the correct answer.

D.3 Output Validation Rules

We enforce structured outputs through explicit format specifications in the prompt. The expected response format is:

```
1. Need help: Yes/No
2. Intention category: [category]
```

Response parsing employs regex-based extraction to handle format variations:

- **Primary pattern matching**: We first search for explicit format adherence using the pattern `Need help: (Yes|No)`.
- **Fallback heuristics**: For responses that deviate from the template, we scan the raw response for keywords indicating the prediction.
- **Method-specific parsing**: For CoT responses, we additionally extract predictions from Step 3 of the reasoning chain when the final summary is malformed.

When the model's response does not conform to the expected format, we apply cascading rules: first searching for explicit markers, then scanning for category keywords with preference for later mentions (as CoT responses typically place final answers at the end). This robust parsing strategy ensures reliable evaluation even when models produce verbose or partially-formatted outputs.

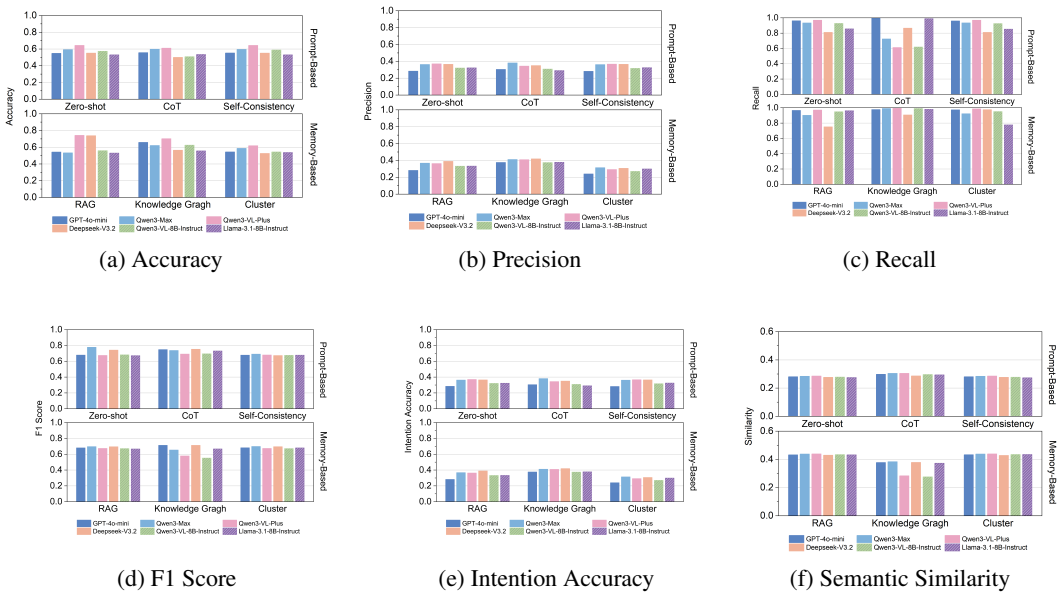
E Full Results

E.1 Base Result & Memory-based Methods Result

This section presents the complete evaluation results comparing base prompt-based methods (Zero-shot, CoT, Self-Consistency) with memory-augmented approaches (RAG, Knowledge Graph and Cluster). Figure 6 shows the performance across all six evaluation metrics: (a) Accuracy, (b) Precision, (c) Recall, and (d) F1 Score for the *When to Assist* task, and (e) Intention Accuracy and (f) Semantic Similarity for the *How to Assist* task. Each subplot compares different methods across multiple LLM backbones, demonstrating the effectiveness of incorporating long-term user context through memory mechanisms.

E.2 Context Time Window Length Ablation

This section presents the ablation study on the impact of historical context length. We evaluate model performance across different time window sizes ranging from 30 seconds to 10 minutes. Figure 7

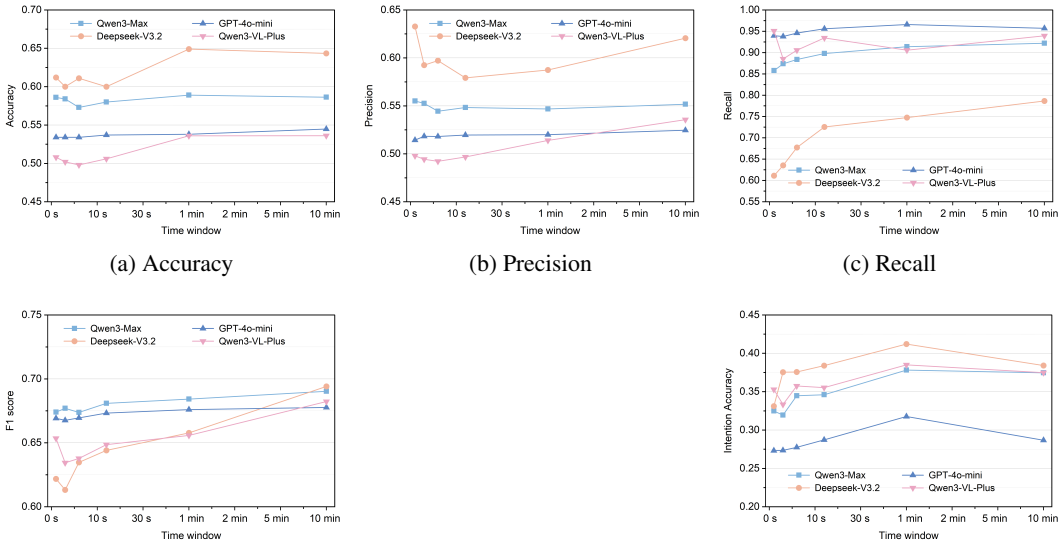


(a) Accuracy (b) Precision (c) Recall

(d) F1 Score (e) Intention Accuracy (f) Semantic Similarity

Figure 6: Base results and memory-based methods comparison across different evaluation metrics for both *When to Assist* (Accuracy, Precision, Recall, F1 Score) and *How to Assist* (Intention Accuracy, Semantic Similarity) tasks.

shows the results: (a) Accuracy, (b) Precision, (c) Recall, and (d) F1 Score for the *When to Assist* task, and (e) Intention Accuracy for the *How to Assist* task. The results demonstrate that longer context windows generally improve performance, with diminishing returns observed beyond the 5-minute mark, suggesting that a 5-minute context window provides an effective balance between capturing sufficient behavioral context and computational efficiency.



(a) Accuracy (b) Precision (c) Recall

(d) F1 Score (e) Intention Accuracy

Figure 7: Full evaluation results across different time window sizes (from 30s to 10m) for both *When to Assist* and *How to Assist* tasks.

E.3 Real-world vs. Synthetic Training Data

This section presents the complete results comparing models fine-tuned on real-world data versus synthetic data. Table 5 shows the performance of LLaMA-3.1-8B-Instruct and Qwen3-VL-8B-Instruct under different fine-tuning strategies (SFT and LoRA) with both data sources. The results demonstrate that real-world data consistently outperforms synthetic data across all metrics, highlighting the unique value of authentic human interaction patterns for training proactive assistance systems.

Table 5: Impact of training data source on open-source models (Full Results). We compare Zero-shot baseline with models fine-tuned on real-world vs. synthetic data using SFT and LoRA. Abbreviations: Acc.=Accuracy, Pre.=Precision, Rec.=Recall, Int. Acc.=Intention Accuracy, Sem. Sim.=Semantic Similarity. Best results per model are **bolded**.

Method	Data	Accuracy	Precision	Recall	F1 Score	Intention Accuracy	Semantic Similarity
LLaMA-3.1-8B-Instruct							
Zero-shot	–	57.3%	54.7%	85.7%	66.7%	32.3%	0.275
SFT	Synthetic	62.1%	58.3%	86.2%	70.2%	34.8%	0.312
	Real-world	**74.0%**	**71.2%**	**87.4%**	**78.5%**	**42.1%**	**0.385**
LoRA	Synthetic	60.5%	57.1%	85.8%	68.9%	33.6%	0.298
	Real-world	71.2%	68.5%	86.1%	76.3%	40.5%	0.372
Qwen3-VL-8B-Instruct							
Zero-shot	–	51.7%	50.9%	94.4%	66.1%	35.3%	0.276
SFT	Synthetic	54.8%	52.6%	92.8%	67.8%	36.2%	0.295
	Real-world	**63.5%**	**60.2%**	**89.5%**	**72.4%**	**41.8%**	**0.358**
LoRA	Synthetic	53.2%	51.8%	93.2%	67.1%	35.9%	0.288
	Real-world	61.8%	58.9%	90.1%	71.2%	40.2%	0.345

E.4 Detailed outputs of LLM

Table 6: Conditional accuracy of each answer, derived from the same Precision and Recall.

Model	Method	P(said help | right) = Precision	P(said no | right) = NPV
GPT-4o-mini	Zero-shot	52.7%	78.2%
	CoT	55.6%	97.6%
	Self-Consistency	52.8%	78.0%
Qwen3-Max	Zero-shot	55.5%	79.2%
	CoT	59.6%	64.9%
	Self-Consistency	55.7%	79.8%
Deepseek-V3.2	Zero-shot	60.8%	71.6%
	CoT	60.9%	76.8%
	Self-Consistency	60.8%	71.6%
Qwen3-VL-Plus	Zero-shot	51.6%	75.0%
	CoT	54.9%	56.2%
	Self-Consistency	51.7%	75.8%
Llama-3.1-8B	Zero-shot	54.7%	67.0%
	CoT	50.4%	72.1%
	Self-Consistency	56.7%	70.3%
Qwen3-VL-8B	Zero-shot	50.9%	61.5%
	CoT	32.7%	43.9%
	Self-Consistency	51.8%	66.8%

As demonstrated in Table 7, the large language models do not adopt an "all positive" prediction strategy. The table explicitly notes that the evaluation is conducted under a "$\sim$50% positive base rate." If the models were simply predicting every sample as positive (i.e., "Help"), the negative prediction metrics—*No & correct* (TN) and *No & wrong* (FN)—would be exactly 0% across all rows. However,

Table 7: Share of all events falling into each outcome, under a ∼50% positive base rate. Derived from the pooled Precision and Recall in Table 2. Best per column **bolded**.

Model	Method	Help & correct (TP)	No & correct (TN)	Help & wrong (FP)	No & wrong (FN)
Closed-Source Models					
GPT-4o-mini	Zero-shot	48.1%	6.8%	43.2%	1.9%
	CoT	**49.8%**	10.3%	39.7%	**0.3%**
	Self-Consistency	48.0%	7.1%	42.9%	2.0%
Qwen3-Max	Zero-shot	46.7%	12.6%	37.4%	3.3%
	CoT	36.3%	**25.4%**	**24.6%**	13.8%
	Self-Consistency	46.8%	12.8%	37.2%	3.3%
Deepseek-V3.2	Zero-shot	40.6%	23.9%	26.1%	9.5%
	CoT	43.3%	22.2%	27.8%	6.7%
	Self-Consistency	40.6%	23.9%	26.1%	9.5%
Qwen3-VL-Plus	Zero-shot	48.5%	4.5%	45.5%	1.5%
	CoT	30.7%	24.8%	25.2%	19.4%
	Self-Consistency	48.5%	4.7%	45.3%	1.5%
Open-Source Models					
Llama-3.1-8B	Zero-shot	42.9%	14.5%	35.5%	7.2%
	CoT	**49.5%**	1.3%	48.7%	0.5%
	Self-Consistency	42.7%	17.4%	32.6%	7.4%
Qwen3-VL-8B	Zero-shot	47.2%	4.5%	45.5%	2.8%
	CoT	8.6%	**32.4%**	17.6%	41.4%
	Self-Consistency	46.8%	6.5%	43.6%	3.2%

the empirical results reveal significant non-zero values for TN and FN across every model and method. For instance, GPT-4o-mini under the Zero-shot setting yields a TN of 6.8% and an FN of 1.9%. Even the most positively biased configuration, Llama-3.1-8B with CoT (which predicts positive in 98.2% of cases, combining TP and FP), still retains 1.3% TN and 0.5% FN. Conversely, several models exhibit a strong negative prediction tendency; for example, Qwen3-VL-8B with CoT predicts negative in 73.8% of cases (TN 32.4% + FN 41.4%). The substantial variation in prediction distributions across different models and methods indicates that they form distinct decision boundaries based on input features rather than blindly outputting a single class. Therefore, the "all positive" hypothesis is firmly rejected by the empirical evidence.

F Ablations

F.1 Ablations 1: Impact of Agent Reasoning Strategies

To explore the potential of advanced reasoning in proactive assistance, we evaluate different strategies including: (1) **Zero-shot**, the baseline approach that generates proactive decisions directly from input user information without explicit reasoning processes; (2) **Chain-of-Thought (CoT)**, which elicits step-by-step reasoning; and (3) **Self-Consistency (SC)**, which aggregates multiple inference paths.

We observe that: (1) SC is the most reliable prompting method, offering consistent performance compared to the baseline. For example, as shown in Table 8, in Qwen3-VL-8B-Instruct (Text-only), while CoT suffers a drastic performance drop (F1 Score: 22.4%), SC maintains robust performance (F1 Score: 66.7%), closely matching and slightly outperforming the Zero-shot baseline (F1 Score: 66.1%), effectively mitigating the volatility seen in complex reasoning chains; (2) Interestingly, CoT prompting often yields lower performance than zero-shot. This aligns with recent findings that CoT degrades performance on tasks involving implicit pattern recognition rather than explicit logical deduction Liu et al. (2025); Zheng et al. (2025). Our analysis reveals that CoT amplifies models' inherent behavioral tendencies: in Deepseek-V3.2 and LLaMA3.1-8B, CoT shifts decision boundaries toward aggressive triggering (higher Recall, lower Accuracy), while in Qwen3-VL-8B, it induces excessive conservatism (Recall drops from 0.944 to 0.171). We further observe that CoT tends to overthink simple scenarios, imagining future problems rather than assessing what the user

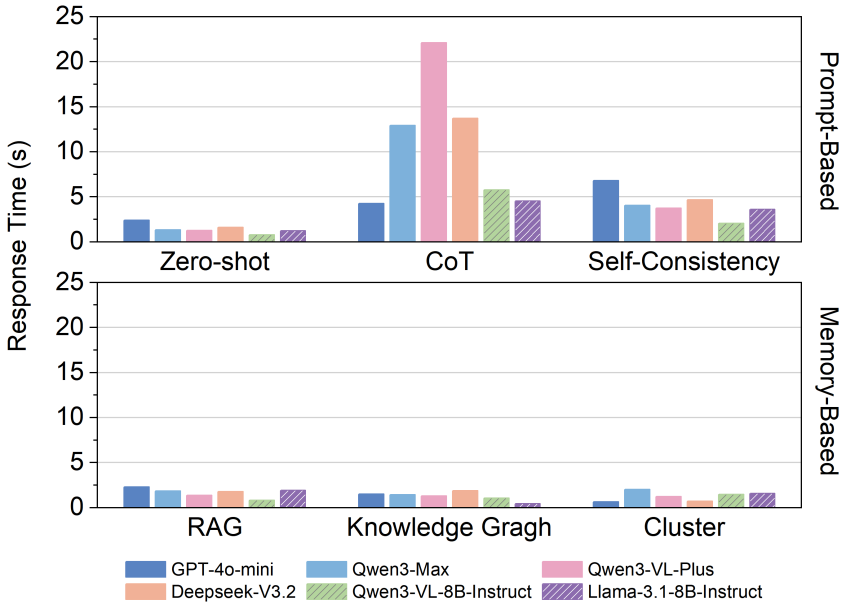


Figure 8: **Inference latency comparison across methods.** Response time (in seconds) for different models using prompt-based methods (top) and memory-based methods (bottom).

actually needs in the present. As illustrated in Figure 9, given a user simply browsing multiple tabs, zero-shot correctly predicts no assistance is needed. Under CoT, however, the model constructs unfounded reasoning about information overload and hypothetical needs to compare page contents, ultimately producing an incorrect prediction. Consequently, for proactive assistance systems where balancing false alarms and coverage is critical, zero-shot or Self-Consistency prompting remains the more robust choice.

F.2 Ablations 2: Impact of Input Modalities

To investigate whether visual context improves proactive assistance, we evaluate model performance across two input modalities: **Multi-modal** and **Text-only**. In the Multi-modal setting, the model receives the raw screen screenshot, combined with the user's historical interaction data and profile information. In contrast, the Text-only setting replaces the visual screenshot with its textual representation, extracted via Optical Character Recognition (OCR), while retaining the same user history and profile context.

We observe that: (1) Surprisingly, integrating visual information does not consistently improve performance and, in many cases, leads to degradation. For instance, in Qwen3-VL-Plus (Table 8), the Multi-modal input yields lower Accuracy (50.6% vs. 53.0%), F1 Score (65.6% vs. 67.4%), and Precision (50.3% vs. 51.6%) compared to the Text-only baseline in the Zero-shot setting. A similar trend is observed in GPT-4o-mini, where Multi-modal accuracy drops to 52.5% from 54.9% (Text-only); (2) Text-only models demonstrate greater stability and efficiency. Across most models and prompting strategies (e.g., Qwen3-VL-8B-Instruct with SC), Text-only inputs achieve comparable or identical F1 Scores (66.7%) to their Multi-modal counterparts, suggesting that current VLMs may struggle to effectively extract actionable proactive cues from complex GUI screenshots, or that the essential context is already sufficiently captured by the text logs.

F.3 Ablations 3: Inference Latency of Different Methods

For real-world deployment of proactive assistance systems, inference latency is a critical factor, as excessive delays can diminish user experience and reduce the practical utility of timely interventions. We measure the average response time across all evaluated methods and models, as shown in Figure 8.

Table 8: Performance of Multi-modal Models on When to Offer Assistance (Text-only vs. Multi-modal Inputs)

Input	Metric	Prompt-based Methods			Memory-based Methods		
		Zero-shot	CoT	SC	RAG	KG	Cluster
Close-source Models							
GPT-4o-mini							
Text-only	Accuracy	54.9%	55.7%	55.2%	54.3%	65.9%	54.4%
	Precision	52.7%	55.6%	52.8%	52.3%	60.5%	52.0%
	Recall	96.2%	99.5%	96.0%	96.6%	97.7%	97.4%
	F1 Score	68.1%	71.3%	68.2%	67.9%	74.8%	67.8%
	Intention Acc.	28.4%	30.5%	28.2%	28.2%	37.6%	24.0%
	Semantic Sim.	0.280	0.298	0.280	0.432	0.378	0.433
Multi-modal	Accuracy	52.5%	49.8%	53.0%	67.3%	62.2%	54.0%
	Precision	51.3%	49.7%	51.6%	60.7%	57.0%	52.0%
	Recall	96.0%	99.8%	95.2%	97.1%	98.0%	97.3%
	F1 Score	66.9%	66.4%	66.9%	74.7%	72.1%	67.8%
	Intention Acc.	34.0%	33.4%	33.3%	29.3%	40.4%	23.1%
	Semantic Sim.	0.282	0.299	0.279	0.433	0.376	0.432
Qwen3-VL-Plus							
Text-only	Accuracy	53.0%	53.5%	53.1%	53.3%	55.7%	53.7%
	Precision	51.6%	54.9%	51.7%	51.8%	53.0%	51.9%
	Recall	97.0%	61.3%	97.0%	97.1%	99.4%	98.6%
	F1 Score	67.4%	57.9%	67.4%	67.5%	69.2%	68.0%
	Intention Acc.	37.1%	34.4%	36.7%	36.3%	40.9%	29.2%
	Semantic Sim.	0.286	0.305	0.286	0.439	0.284	0.439
Multi-modal	Accuracy	50.6%	46.7%	50.5%	55.7%	63.9%	53.4%
	Precision	50.3%	45.8%	50.3%	53.2%	62.5%	51.9%
	Recall	94.2%	35.9%	93.4%	95.0%	99.7%	97.3%
	F1 Score	65.6%	40.3%	65.4%	68.2%	76.8%	67.7%
	Intention Acc.	38.2%	38.0%	38.5%	38.1%	44.2%	31.0%
	Semantic Sim.	0.286	0.296	0.285	0.439	0.384	0.438
Open-source Models							
Qwen3-VL-8B-Instruct							
Text-only	Accuracy	51.7%	41.0%	52.9%	55.9%	56.5%	52.6%
	Precision	50.9%	32.7%	51.8%	53.3%	53.5%	51.4%
	Recall	94.4%	17.1%	93.6%	94.9%	99.1%	97.6%
	F1 Score	66.1%	22.4%	66.7%	68.2%	69.5%	67.3%
	Intention Acc.	35.3%	34.1%	35.7%	33.2%	37.5%	27.0%
	Semantic Sim.	0.276	0.277	0.274	0.434	0.277	0.435
Multi-modal	Accuracy	51.7%	41.0%	52.9%	65.9%	54.2%	52.2%
	Precision	50.9%	32.7%	51.8%	60.3%	52.2%	51.2%
	Recall	94.4%	17.1%	93.6%	93.7%	99.0%	98.2%
	F1 Score	66.1%	22.4%	66.7%	73.4%	68.4%	67.3%
	Intention Acc.	35.3%	34.1%	35.7%	32.5%	41.6%	28.0%
	Semantic Sim.	0.276	0.295	0.274	0.434	0.376	0.436

We observe that: (1) Most methods achieve real-time or near-real-time inference. Zero-shot prompting demonstrates the lowest latency, with most models responding within 5 seconds, making it highly suitable for latency-sensitive applications. Memory-based methods (RAG, Knowledge Graph, Clustering) also maintain low latency (<2 seconds), as the retrieval and reasoning overhead is minimal compared to generation; (2) **Chain-of-Thought (CoT) substantially increases inference time.** Across all

Table 9: **High visual complexity** (top-100 by S_{vis}). Help accuracy and intention accuracy on valid intent predictions; n counts test cases with parseable intent on *How to Assist*.

Model	Modality	Help Acc.	Intent Acc.	Intent valid n
DeepSeek	Text-only	0.61	0.3077	52
Qwen3-VL-Plus	Vision	0.36	0.1111	9
Qwen3-VL-8B-Instruct	Vision	0.41	0.2222	9

Table 10: **Low visual complexity** (bottom-100 by S_{vis}). Same metrics as Table 9.

Model	Modality	Help Acc.	Intent Acc.	Intent valid n
DeepSeek	Text-only	0.59	0.3333	57
Qwen3-VL-Plus	Vision	0.59	0.4103	39
Qwen3-VL-8B-Instruct	Vision	0.43	0.4000	20

models, CoT introduces significant latency overhead due to the explicit multi-step reasoning process. For instance, Qwen3-VL-Plus requires approximately 22 seconds, while Deepseek-V3.2 and Qwen3-Max take around 13-14 seconds, which is an order of magnitude slower than Zero-shot. This latency penalty, combined with CoT's inconsistent performance improvements (Section F.1), suggests that CoT may not be the optimal choice for proactive assistance scenarios where rapid response is essential; (3) Self-Consistency exhibits moderate latency (3-7 seconds), as it requires multiple sampling passes. Given its stable performance and acceptable latency trade-off, Self-Consistency represents a reasonable middle ground for applications that can tolerate slightly longer response times.

G Additional Analyses

This section documents supplementary analyses addressing reviewer questions on (i) whether text–vision gaps are driven by OCR quality versus dense-GUI visual reasoning, (ii) sensitivity to the screenshot sensing interval, and (iii) whether embedding-based semantic similarity underestimates helpful but paraphrased assistance.

G.1 Setup: Pixel-Level GUI Visual Complexity Score

Our dataset does not include native UI trees; we therefore score each primary screenshot using **pixel-level** statistics only. For each sample we load the screenshot, convert it to grayscale, and resize so the long edge is 640 pixels. We compute: (i) **normalized grayscale entropy** e: Shannon entropy of the 256-bin intensity histogram, divided by $\ln(256)$; (ii) **edge density** d: mean edge-filter response, scaled to $[0, 1]$; (iii) **high-contrast block ratio** c: the fraction of 16×16 non-overlapping blocks whose grayscale standard deviation exceeds the image-wide median block standard deviation. As an auxiliary signal we use total OCR text length ℓ for the event and define $o = \ln(1 + \ell)$. Each feature is z-scored over all valid-image samples; the composite score is

$$S_{\text{vis}} \;=\; 0.8 \cdot \frac{z_e + z_d + z_c}{3} \;+\; 0.2 \cdot z_o \,, \tag{13}$$

so the ranking emphasizes visual clutter rather than raw text volume. We sort test events by S_{vis} and evaluate the top-100 (visually *complex*) and bottom-100 (*simple*) subsets under the same protocol as the main benchmark (text-only vs. vision inputs, identical prompts and parsing).

On simple layouts, vision models match or exceed text-only baselines on intention accuracy; on complex layouts, vision models degrade sharply and often fail to emit parseable intent labels (small n), indicating that aggregate text–vision gaps are driven primarily by **visually dense GUIs**, not by OCR replacing an easy vision shortcut.

Table 11: Impact of sensing interval t on prediction quality and CPU overhead (Qwen3-VL-Plus, text-only zero-shot, $w = 5$ min).

Interval	F1	Recall	Intent Acc.	Sem. Sim.	CPU
1 s	**67.5%**	**97.0%**	37.1%	**0.286**	11.8%
5 s	66.6%	95.2%	36.7%	0.282	2.3%
10 s	58.8%	87.3%	30.5%	0.267	**1.2%**

Table 12: LLM-as-a-Judge vs. cosine semantic similarity (**Zero-shot**). Judge: Qwen-Max; cosine mean $\pm$ std as reported in the rebuttal analysis.

Model	Judge $\hat{s}$	Cosine sim.
GPT-4o-mini	0.2597	0.2804 ± 0.1338
Qwen3-Max	0.2743	0.2845 ± 0.1382
Deepseek-V3.2	0.2442	0.2763 ± 0.1381
Qwen3-VL-8B-Instruct	0.2476	0.2781 ± 0.1347
Llama-3.1-8B-Instruct	0.2641	0.2752 ± 0.1306
Qwen3-VL-Plus	0.2562	0.2858 ± 0.1377

G.2 Setup: Sensing-Interval Ablation

LifeTrace captures screenshots at $\sim$1 Hz by default; we ablate the **effective sensing interval** $t \in \{1\text{s}, 5\text{s}, 10\text{s}\}$ while holding the historical context window fixed at **5 minutes**. All runs use **Qwen3-VL-Plus** with the **text-only zero-shot** pipeline (OCR replaces pixels) on the same test split. We report F1, recall, intention accuracy, semantic similarity (embedding cosine, as in the main paper), and mean **CPU utilization** on an Apple M2 laptop during end-to-end evaluation.

Short intervals preserve timing-critical cues but increase host CPU load; 5 s offers much lower CPU with minor metric drops, while 10 s begins to miss rapid context shifts.

G.3 Setup: LLM-as-a-Judge for Semantic Alignment

Embedding cosine similarity can penalize responses that are helpful but phrased differently from the user's literal query. We therefore add an **LLM-as-a-Judge** protocol using **Qwen-Max** as an independent judge. For each **positive** test sample (ground-truth assistance needed), the judge receives the model's generated assistance text and the user's **actual** query from the log, together with brief activity context, and assigns an integer score $s \in \{1, 2, 3, 4, 5\}$ reflecting semantic alignment from irrelevant to precisely on-target. Scores are mapped to $[0, 1]$ by $\hat{s} = (s - 1)/4$. Judgments use temperature $T = 0$ for repeatability. We report judge scores alongside the main paper's embedding cosine similarity under three prompting strategies (Zero-shot, CoT, Self-Consistency), for the same model set as in the primary *How to Assist* experiments.

Under CoT and Self-Consistency, several models achieve substantially higher judge scores than cosine similarity, suggesting that part of the low cosine overlap reflects metric conservatism rather than useless generations.

G.4 Summary

We briefly summarize the takeaways from the three analyses above. **(1) Visual complexity.** When GUI layouts are simple, vision models can match or beat text-only models; when layouts are visually dense, vision performance drops sharply and valid intent outputs become rare. This pattern is easier to explain as a limit of current VLMs on cluttered screens than as a side effect of high-quality OCR. **(2) Sensing rate.** A 1 s interval gives the strongest metrics; moving to 5 s cuts CPU use substantially with only small score changes, whereas 10 s clearly weakens timing-related metrics, so the default $\sim$1 Hz setting is reasonable for quality-first evaluation. **(3) LLM-as-a-Judge.** Under CoT and Self-Consistency, judge scores often rise well above embedding cosine similarity, so low cosine in the main tables should not be read as "unhelpful output" by itself; the two views are complementary.

Table 13: LLM-as-a-Judge vs. cosine semantic similarity (**CoT**).

Model	Judge $\hat{s}$	Cosine sim.
GPT-4o-mini	0.3868	0.2984 ± 0.1389
Qwen3-Max	0.3230	0.3048 ± 0.1432
Deepseek-V3.2	0.2476	0.2866 ± 0.1512
Qwen3-VL-Plus	0.3211	0.3046 ± 0.1414
Qwen3-VL-8B-Instruct	0.3497	0.2965 ± 0.1375
Llama-3.1-8B-Instruct	0.3565	0.2939 ± 0.1342

Table 14: LLM-as-a-Judge vs. cosine semantic similarity (**Self-Consistency**).

Model	Judge $\hat{s}$	Cosine sim.
GPT-4o-mini	0.3923	0.2804 ± 0.1338
Qwen3-Max	0.3980	0.2845 ± 0.1382
Deepseek-V3.2	0.2500	0.2763 ± 0.1381
Qwen3-VL-Plus	0.2984	0.2858 ± 0.1380
Qwen3-VL-8B-Instruct	0.2488	0.2779 ± 0.1343
Llama-3.1-8B-Instruct	0.2613	0.2736 ± 0.1294

H CoT Failure Cases

We observe that CoT prompting yields mixed results depending on model capacity. For larger models such as Deepseek-V3.2, CoT improves the F1 score from 69.5% to 71.3% on timing prediction. However, for smaller open-source models, CoT can be detrimental. For example, Qwen3-VL-8B-Instruct experiences a dramatic performance drop, with accuracy falling from 51.7% to 41.0%. This aligns with recent findings that CoT degrades performance on tasks involving implicit pattern recognition rather than explicit logical deduction Liu et al. (2025); Zheng et al. (2025). Our analysis reveals that CoT amplifies models' inherent behavioral tendencies: in Deepseek-V3.2 and LLaMA3.1-8B, CoT shifts decision boundaries toward aggressive triggering (higher Recall, lower Accuracy), while in Qwen3-VL-8B, it induces excessive conservatism (Recall drops from 0.944 to 0.171). We further observe that CoT tends to overthink simple scenarios, imagining future problems rather than assessing what the user actually needs in the present, as illustrated in Figure 9. On the *How to Assist* task, CoT provides modest improvements in semantic similarity (e.g., Qwen3-Max improves from 0.285 to 0.305), indicating that structured reasoning helps models better articulate assistance content.

As is demonstrated in Fig. 9, given a user simply browsing multiple tabs, zero-shot correctly predicts no assistance is needed. Under CoT, however, the model constructs unfounded reasoning about information overload" and hypothetical needs to compare page contents," ultimately producing an incorrect prediction. Consequently, for proactive assistance systems where balancing false alarms and coverage is critical, zero-shot or Self-Consistency prompting remains the more robust choice.

Zero-shot Prediction		
Output	"Need help: **No**; Intent category: **Uncategorized**."	✓ **Correct**

Chain-of-Thought Prediction		
Step	**Reasoning Path**	**Assessment**
S1	"The user is currently using the Microsoft Edge browser with multiple tabs open, including a page related to Google Gemini. The user has been active in this browser during the past five minutes."	Observed context
S2	"The user may be browsing information about Google Gemini or conducting online activities such as research or study. However, since no explicit text content is available, the exact activity cannot be determined."	Speculation with uncertainty
S3	"Because the user has multiple pages open, they may experience information overload, need to quickly locate specific information, or compare content across tabs. Therefore, the user may require assistance."	Unfounded inference
S4	"The user's intent is likely *Information Query*, as they may be searching for Gemini-related information or comparing multiple pages."	Intent inferred from fabrication
Final	"Need help: **Yes**; Intent: **Information Query**."	✗ **Incorrect**

Figure 9: Illustrative false-positive case of Chain-of-Thought (CoT) reasoning in proactive help prediction. **Context:** The user is browsing multiple tabs in Microsoft Edge. **Ground truth:** No assistance is required. While the zero-shot model directly outputs the correct decision, the CoT model progressively introduces hypothetical user difficulties (underlined in Step S3), leading to an incorrect prediction of help need and intent.