

SAPA: a Situation-Augmented Personalized web Agent

Qianyue Li Bin Ji* Xiaodong Liu* Shasha Li Xiaopeng Li

Yueyu Wang* Xinran Hong Jun Ma Jie Yu*

National University of Defense Technology

Changsha, China

{li_qianyue, jibin, liuxiaodong, shashali, xiaopengli,
wangyueyu, hongxinran, majun, yj}@nudt.edu.cn

Abstract

Large Language Model (LLM)-based agents have achieved promising performance in web task planning. However, they primarily focus on task completion rather than preference modeling, leading to imprecise personalization. This can be attributed to two facts: (1) Contextual scarcity. Vague instructions and insufficient background of users' inputs make it difficult to accurately identify users' intents. (2) Preference unpredictability. It's hard for a single response path to handle users' sudden intent changes. To address these issues, we propose **SAPA**, a **Situation-Augmented Personalized web Agent**. Specifically, a *Situational Intent Grounding* module is proposed to enrich the contexts of users' inputs by incorporating historical records and candidate constraints, effectively alleviating the contextual scarcity. Then, we propose a *Task-Driven Semantic Expansion* module to explore diverse response paths and reduce preference unpredictability in turn, which is achieved by employing a test-time expansion strategy. We construct SAPA with representative LLMs and conduct extensive experiments on PersonalWAB and Amazon Reviews datasets. On PersonalWAB, SAPA achieves new state-of-the-art performance with improvements of +14.0%, +8.8%, and +1.3% in Overall score on personalized recommendation, search, and question answering, respectively. We further evaluate on three single-category scenarios from Amazon Reviews (Office Products, Beauty and Personal Care, Electronics), where SAPA consistently improves Overall scores by 16%–18% with gains across all task types. Ablation studies and case analyses further confirm that the gains stem from jointly addressing contextual scarcity and preference unpredictability. Our code is available at <https://github.com/QianyLi/SAPA>.

1 Introduction

Large Language Model (LLM)-based agents have demonstrated remarkable potential in autonomous web navigation. Some existing studies, such as WebGPT (Nakano et al., 2021) and WebCPM (Qin et al., 2023), have established the feasibility of autonomous browsing for open-domain question answering. Some other studies have extended these capabilities to more complex, goal-oriented environments. For instance, agents in WebShop (Yao et al., 2022) manage sequential decision-making in e-commerce interfaces, generalist agents like Mind2Web (Deng et al., 2023) handle diverse cross-domain tasks. Recent studies leverage LLMs' conversational capabilities to facilitate multi-turn user dialogues and enhance conversational task response (Deng et al., 2024; Chen et al., 2025). While existing web agents have achieved promising performance on general task completion, they primarily focus on whether tasks are successfully executed, with limited attention to modeling user preferences. Consequently, they still fall short in precise personalizations, which can be attributed to **contextual scarcity** and **preference unpredictability**. Specifically, the instructions and background of users' inputs are often ambiguous and insufficient, and the commonly used single response struggles to handle users' sudden intent changes.

In light of the above issues, we propose **SAPA**, a **Situation-Augmented Personalized web Agent**. To be specific, SAPA includes a **Situational Intent Grounding** module to alleviate the contextual scarcity problem, which maps vague user references to concrete task entities by leveraging historical records and filtered candidate pools. Moreover, SAPA incorporates a **Task-Driven Semantic Expansion** module to reduce preference unpredictability, which is achieved by employing test-time n -sampling to explore diverse response paths, thereby ensuring robust coverage even when user prefer-

* Corresponding author.

ence deviates from historical patterns.

To validate the effectiveness of SAPA, we conduct extensive experiments on **PersonalWAB** (Cai et al., 2025), the standard personalized web agent benchmark, where SAPA achieves up to **+14.0%, +8.8%, and +1.3%** improvements on recommendation, search, and review tasks, establishing new state-of-the-art results. To further demonstrate that SAPA is not limited to a specific benchmark, we evaluate on three additional personalized agent scenarios constructed from single-category Amazon subsets (Hou et al., 2024) (**Office, Beauty, Electronics**), where SAPA consistently delivers **16%–18% Overall improvements**. Ablation studies confirm that the observed gains stem from jointly addressing contextual scarcity (via Grounding) and preference unpredictability (via task-adaptive expansion).

In summary, our contributions are three-fold: (1) we identify two key bottlenecks in personalized web agents, namely *contextual scarcity* and *preference unpredictability*, which respectively arise from the input and output sides of personalized tool invocation; (2) we propose **SAPA**, a unified framework that combines *Situational Intent Grounding* for training with *Task-Driven Semantic Expansion* for inference across three web tasks; and (3) we conduct extensive experiments on PersonalWAB and three Amazon datasets, showing consistent improvements over strong baselines.

2 Related Work

LLM-based Web Agents. Recent years have witnessed a surge in LLM-based web agents. Benchmarks such as Mind2Web (Deng et al., 2023) and WebArena (Zhou et al., 2024) have established realistic platforms to evaluate agent performance across tasks ranging from information retrieval to service booking. Advanced frameworks like WebVoyager (He et al., 2024) and SteP (Sodhi et al., 2024) further enhance these capabilities by integrating multi-modal perception and hierarchical planning. Despite these advancements in planning and reasoning capabilities, existing agents primarily focus on general task execution without modeling user preferences for web content presentation. Consequently, they struggle to ground vague user instructions and align outputs with individual user preferences.

Personalized Web Navigation. Integrating user history into LLMs has been explored in specialized

domains like recommendation (Wang et al., 2024). In web navigation, PUMA (Cai et al., 2025) introduced an agentic workflow for personalized browsing by incorporating memory-augmented retrieval. However, PUMA’s reliance on matching current intent with specific historical behaviors limits its capacity to handle situational shifts. Meanwhile, AdaPA (Nie et al., 2025) attempts to capture dynamic preferences by mixing logic and arithmetic during inference, yet it still operates within a deterministic framework that struggles to accommodate sudden changes in user intent. Our work departs from these history-constrained paradigms by utilizing situational metadata for intent grounding and test-time expansion for proactive exploration, allowing the agent to move beyond the constraints of historical patterns when user requirements shift unexpectedly.

Inference-time Scaling. Reasoning paradigms such as Self-Refine (Madaan et al., 2023) and Tree-of-Thoughts (Yao et al., 2023a) enhance performance through iterative refinement, but suffer from error propagation in smaller models like Llama-2 (Touvron et al., 2023), where inaccurate self-critiques accumulate into hallucinations. While increasing inference-time computation has proven effective (Snell et al., 2024), uniform expansion strategies fail to adapt to varying task complexity. SAPA addresses this through task-adaptive parallel sampling that dynamically adjusts exploration breadth based on task characteristics and intent uncertainty. Unlike recursive reasoning that propagates errors sequentially, or preference alignment methods like DPO (Rafailov et al., 2023) that risk mode collapse, SAPA maintains broader exploration while avoiding error accumulation. This ensures responsiveness to intent instability and reachability of correct responses even when requirements deviate from historical patterns.

3 Methodology

3.1 Problem Formulation

Given a personalized web task, let $\mathcal{Q}$ be the user instruction, $\mathcal{H}$ the historical behavior metadata, $\mathcal{C}$ the situational candidate product pool, and $\mathcal{P}$ the global product database. SAPA optimizes task parameters Y to fulfill intent $\mathcal{I}$ via two modules:

- (1) *Situational Intent Grounding*: Anchoring the

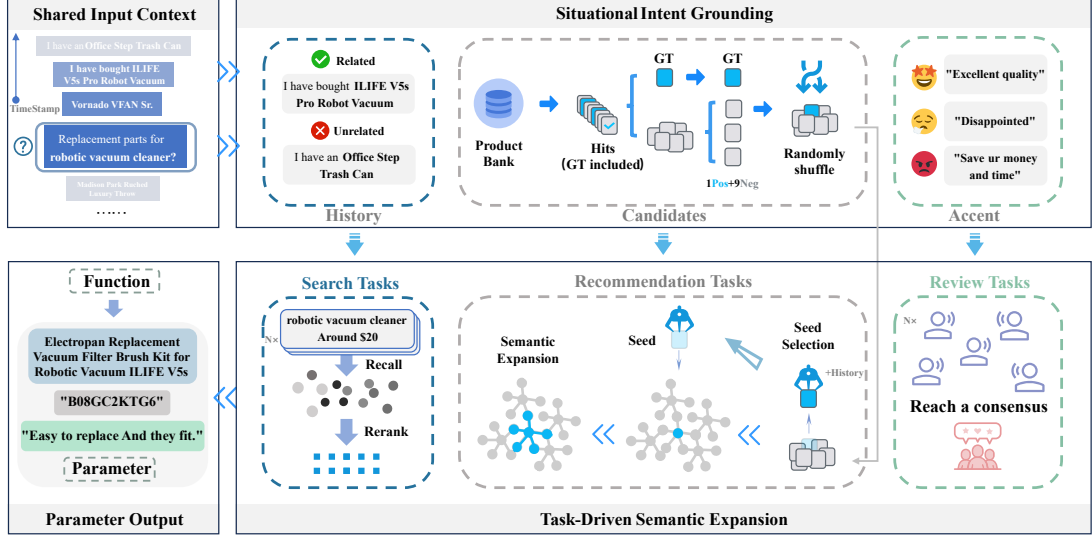


Figure 1: Overview of the proposed SAPA framework. *Situational Intent Grounding* aligns vague intent using historical metadata and candidates. *Task-Driven Semantic Expansion* leverages test-time scaling and refinement to accommodate intent volatility and model stochasticity.

latent intent $\mathcal{I}$ using situational priors:

$$\mathcal{I} = \begin{cases} f_{\text{ground}}(\mathcal{Q}, \mathcal{H}, \mathcal{C}), & \text{if recommendation,} \\ f_{\text{ground}}(\mathcal{Q}, \mathcal{H}), & \text{for search and review.} \end{cases} \quad (1)$$

This formulation ensures that for recommendation, the intent is anchored within the specific candidate space $\mathcal{C}$, while for generative tasks, the intent is distilled directly from the interaction between the instruction and historical context.

(2) *Task-Driven Semantic Expansion*: Exploring the semantic space to identify the optimal path Y^* :

$$Y^* = \text{Ref}(\{Y_i\}_{i=1}^n), \quad Y_i \sim P(Y | \mathcal{I}, \mathcal{P}), \quad (2)$$

where n is the number of test-time samples and $\text{Ref}(\cdot)$ denotes the selection strategy to refine the final output.

3.2 Situational Intent Grounding

User intent often requires refinement by integrating both historical patterns and current situational factors. To enable the LLM to distill intent in such contexts, SAPA augments the situational input with $\mathcal{H}$ and $\mathcal{C}$, thereby aligning the model's reasoning with the user's actual requirements.

Input Construction. For a given instruction $\mathcal{Q}$, we construct a situational prompt by augmenting the original query with two key components:

(1) *Historical Metadata $\mathcal{H}$* : This includes filtered records of the user's past behaviors and long-term preferences, providing the necessary priors to interpret personalized needs. To reduce noise,

we apply a task-specific extraction function Φ to the user's historical records $\mathcal{H}$: *i)* For search, we extract product attributes including title, category, price, and store name; *ii)* For recommendation, we retain the product title, category, and parent ASIN (ID); *iii)* For review tasks, only the user's past ratings and comments are preserved. Such historical contexts can be defined as:

$$\mathcal{H}_{ctx} = \text{Extract}(\text{TopK}(\mathcal{H}, \text{sim}(\mathcal{Q}, h_j), K), \Phi), \quad (3)$$

where $\text{sim}(\cdot)$ denotes the similarity between the current query $\mathcal{Q}$ and each historical record $h_j \in \mathcal{H}$, and K is the retrieval threshold.

(2) *Situational Candidate Pool $\mathcal{C}$* (utilized only for recommendation): This represents the set of entities available in the current situation. Specifically, for each training instance, given the global product sequence $\mathcal{P}$, we construct a candidate set $\mathcal{C}_{train}$ containing n items. This set consists of two components, including ground truth (GT) denoted as p^+ and a set of hard negative samples $\mathcal{P}^-$. Subsequently, the set $\mathcal{C}_{train} = \{p^+\} \cup \mathcal{P}^-$ is randomly shuffled and input into the model. For evaluation, we simply select the top-10 semantically relevant products and shuffle them as candidates $\mathcal{C}_{test}$ according to the instructions to simulate a real-world scenario.

By providing these multi-source signals, SAPA constrains the agent's reasoning scope, allowing it to map latent preferences to specific task targets within the environment.

SFT for Intent Anchoring. We fine-tune the backbone LLM to distill situational signals into a grounded target intention $\mathcal{I}$. The specific form of $\mathcal{I}$ varies by task: (1) In recommendation, $\mathcal{I}$ is a discrete *seed entity* ($\mathcal{I} \in \mathcal{C}$) that anchors the user’s product interest. (2) In search, $\mathcal{I}$ is a set of *refined keywords* that distill the key information need from the historical context. (3) In review tasks, $\mathcal{I}$ is a set of *prototype reviews* aligned with the user’s historical feedback style. Even when the original query $\mathcal{Q}$ is misleading or inconsistent, these grounded outputs ensure accurate intent alignment, providing a reliable foundation for the subsequent task-driven expansion of response strategies.

3.3 Task-Driven Semantic Expansion

The second module explores the semantic space around the grounded intention $\mathcal{I}$ to accommodate intent fluctuation and response stochasticity. By scaling computation at test-time, SAPA ensures that the optimal response path Y^* remains reachable even when user requirements deviate from historical patterns.

Semantic Exploration. Starting from the anchored intention $\mathcal{I}$, we explore the related semantic solution space. In recommendation, we retrieve entities from $\mathcal{P}$ that are semantically similar to the seed $\mathcal{I}$:

$$N_{neighbors} = \text{Top-K}_{j \in \mathcal{J}}(\text{sim}(\mathcal{I}, p_j)), \quad (4)$$

$$R = \mathcal{I} \cup N_{neighbors}, \quad (5)$$

where $p_j \in \mathcal{P}$. For search and review tasks, we perform n -path sampling $Y_i \sim P(Y | \mathcal{I}, \mathcal{P})$ with an increased decoding temperature to explore a broader range of potential responses.

Task-Specific Refinement. To identify the most robust response Y^* from the expanded candidates, we employ task-specific selection strategies.

(1) Search Reranking: To avoid potential omission errors, we adopt a hybrid strategy. Let $\mathcal{A}$ be the aggregated candidate set retrieved by the n generated queries:

$$\mathcal{A} = \bigcup_{i=1}^n \text{BM25}(q_i, \mathcal{P}), \quad (6)$$

where $\text{BM25}(q_i, \mathcal{P})$ (Robertson and Zaragoza, 2009) denotes the sparse retrieval from the database $\mathcal{P}$. We identify the optimal result set Y^* by scoring candidates against the original instruction $\mathcal{Q}$:

$$Y^* = \arg \text{top}_{p \in \mathcal{A}}^K \phi(p, \mathcal{Q}). \quad (7)$$

We employ BGE-Reranker-v2-m3 (Chen et al., 2024) as the reranking function $\phi(\cdot)$, which leverages a cross-encoder architecture to capture fine-grained semantic interactions. By reranking the distinct candidates in $\mathcal{A}$ and retaining the Top- K ($K = 10$) results, SAPA effectively filters out retrieval noise and ensures that the final output is precisely aligned with the user’s intent.

(2) Review Centroid Refinement: To resolve potential semantic misalignments, we implement a selection mechanism based on semantic consensus. Let $\mathcal{Y} = \{y_1, \dots, y_n\}$ be the set of n review samples generated in the expansion module. We first encode each sample into a normalized embedding $\mathbf{v}_i$. The optimal review y^* is then identified by maximizing the cumulative similarity within the set:

$$y^* = \arg \max_{y_i \in \mathcal{Y}} \sum_{j=1}^n \text{Sim}(\mathbf{v}_i, \mathbf{v}_j), \quad (8)$$

where $\text{Sim}(\cdot)$ denotes the cosine similarity. By selecting the candidate that achieves the highest consensus score, we effectively identify the central sample. This mitigates the impact of intent drift and model stochasticity, ensuring the final output aligns with the most consistent response observed across multiple sampled paths.

This expansion-and-refinement process allows the agent to transcend the limitations of single-shot generation, providing a reliable solution for complex web navigation.

4 Experiments

4.1 Experimental Setup

Dataset and Tasks. We evaluate SAPA on four datasets to validate effectiveness across diverse compositions. (1) **PersonalWAB** (Cai et al., 2025) is a multi-category benchmark with 35,772 items across five categories and 1,000 users. It contains three task types: search (query generation), recommendation (product ranking), and review (text generation). Following the standard protocol, 80% of interactions serve as historical context, with the remaining 20% split into training (10%) and test (10%), yielding 6,896/2,174 samples per task. (2) **Amazon datasets** are derived from Amazon Reviews 2023 (Hou et al., 2024), focusing on three single-category subsets: Office Products (11,196 items), Beauty & Personal Care (15,167 items), and Electronics (14,596 items). Each contains 1,000 users processed with 10-core filtering. We construct tasks following the same three-task protocol

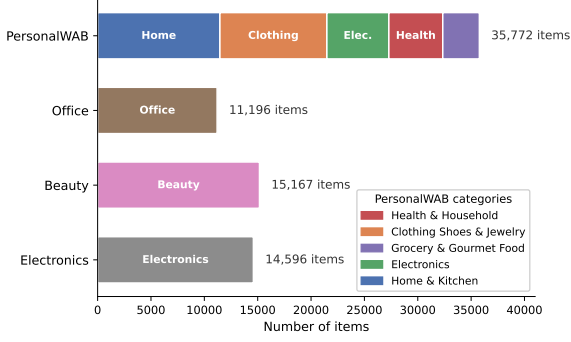


Figure 2: Category distribution and catalog size comparison between PersonalWAB and three single-category Amazon scenarios. PersonalWAB spans five product categories with 35,772 items, while each Amazon scenario focuses on a single category with $\sim 11\text{K}$ – 15K items.

and apply a 76%/24% train/test split (2,280/720 samples per task). SAPA is trained independently on each dataset to assess generalizability across user behavior patterns (cross-category generalists vs. single-category specialists) and catalog scales. Detailed statistics are in Appendix E; Figure 2 illustrates category distributions.

Evaluation Metrics. According to the standard evaluation protocol for personalized agents, we employ task-specific metrics. For Search and Recommend tasks, we use Result Accuracy (*Res. Acc.*), which is a rank-decayed score based on the target product’s position within the top-10 results. Given the rank r of the ground truth item, the score is defined as:

$$\text{Res. Acc.} = \begin{cases} 1 - \frac{r-1}{10}, & \text{if } 1 \leq r \leq 10, \\ 0, & \text{if } r > 10. \end{cases} \quad (9)$$

For the Review task, we assess semantic consistency between the generated text and actual reviews via cosine similarity. We utilize the pre-trained all-MiniLM-L6-v2 (Reimers and Gurevych, 2019) model to compute the embeddings. Finally, an average score is calculated as the arithmetic mean across all tasks to represent the overall performance of the agent.

Baselines. We evaluate SAPA by comparing it against a diverse set of representative baselines. Unless specified otherwise, all methods utilize GPT-4o-mini as the default backbone: (i) Memory Retrieval Methods: These include *No Memory* (instruction only), *Random Memory*, *Last Memory*, and *Relevant Memory* (similarity-based retrieval). (ii) Reasoning and Feedback Mechanisms: We evaluate *ReAct* (Yao et al., 2023b), which implements

a synergy of reasoning and acting, and *Reflexion* (Shinn et al., 2023), which enhances agentic performance through linguistic self-reflection and iterative error correction. (iii) Specialized Personalization Frameworks: We include *RecMind* (Wang et al., 2024) and *InteRecAgent* (Huang et al., 2025), both designed for personalized recommendation. We also compare against *PUMA* (Cai et al., 2025), the state-of-the-art framework for personalized web navigation. PUMA’s performance is reported on both gpt-4o-mini and Llama-2-7b for comparison. (iv) Adaptive Inference Methods: We evaluate *AdaPA* (Nie et al., 2025), which captures dynamic user preferences by mixing logic and arithmetic weights during inference without extra training.

Implementation Details. SAPA uses Llama-2-7B as the backbone, with additional evaluations on Llama-3-8B (Llama Team, 2024), Qwen-2.5-7B (Yang et al., 2025), and Qwen-3.5-9B (Qwen Team, 2026) (Appendix D). SFT is fine-tuned via LoRA (Hu et al., 2022) with learning rate $3e-4$. We set similarity threshold $\tau = 0.5$ for recommendation grounding, sampling count $N = 10$ and temperature $T = 1$ for semantic expansion. Search tasks use BM25 retrieval with BGE Reranker-v2-m3 reranking; review tasks use all-MiniLM-L6-v2 for semantic centroid computation. Experiments run on 8 NVIDIA L40S GPUs. Main results are averaged over three runs with different seeds.

4.2 Main Results

This section reports the experimental results achieved by SAPA on the PersonalWAB benchmark with GPT-4o-mini and Llama-2-7B. For clear comparisons, we analyze the main results from single- and multi-turn perspectives.

4.2.1 Single-turn Track.

Table 1 presents the performance of SAPA and baseline methods on PersonalWAB under the single-turn setting, where the agent has only one opportunity to select the correct function and generate the optimal parameter. We report both Function Accuracy (F. Acc., whether the correct tool is selected) and Result Accuracy (Res. Acc., the rank-decayed score of the target product).

SAPA achieves state-of-the-art performance. SAPA consistently outperforms all baselines across both model scales. Notably, SAPA with Llama-2-7B even surpasses PUMA with the more powerful GPT-4o-mini, demonstrating that architectural innovations in contextual enrichment can

Method	Search		Recommend		Review		Average	
	F. Acc.	Res. Acc.	F. Acc.	Res. Acc.	F. Acc.	Res. Acc.	F. Acc.	Res. Acc.
GPT-4o-mini								
No Memory	1.000	0.647	0.092	0.000	1.000	0.444	0.684	0.355
Random Memory	0.974	0.640	0.296	0.018	0.996	0.442	0.745	0.357
Last Memory	0.937	0.626	0.432	0.028	1.000	0.442	0.782	0.357
Relevant Memory	0.928	0.622	0.492	0.030	1.000	0.443	0.800	0.356
ReAct	0.903	0.605	0.560	0.027	0.996	0.444	0.815	0.350
RecMind	0.981	0.645	0.226	0.017	0.990	0.442	0.721	0.359
AdaPA	0.987	0.654	0.592	0.027	1.000	0.457	0.851	0.367
PUMA	1.000	0.649	0.939	0.048	1.000	0.449	0.979	0.373
SAPA(Ours)	0.989	<u>0.711</u>	0.989	<u>0.167</u>	0.991	0.446	0.990	<u>0.433</u>
Llama-2-7B								
PUMA	0.996	0.652	<u>0.987</u>	0.054	1.000	<u>0.538</u>	<u>0.994</u>	0.406
SAPA(Ours)	<u>0.997</u>	0.740	0.989	0.194	1.000	0.551	0.995	0.487

Table 1: Performance comparison on PersonalWAB benchmark on the **single-turn** track. We report Function Accuracy (F. Acc.) and Result Accuracy (Res. Acc.) across three tasks. Bold numbers indicate the best performance.

Method	Search			Recommend			Review			Average		
	F. Acc.	R. Acc.	Steps	F. Acc.	R. Acc.	Steps	F. Acc.	R. Acc.	Steps	F. Acc.	R. Acc.	Steps
No Memory	0.996	0.656	2.39	0.096	0.000	2.42	1.000	0.446	2.01	0.685	0.358	2.28
Random Memory	0.999	0.680	4.19	0.703	0.042	4.47	1.000	0.448	2.00	0.896	0.380	3.56
Last Memory	0.996	0.676	4.22	0.708	0.045	4.25	1.000	0.449	2.00	0.897	0.381	3.49
Relevant Memory	0.996	0.686	4.23	0.715	0.042	4.56	0.999	0.448	2.00	0.899	0.383	3.60
ReAct	0.996	0.674	4.65	0.218	0.013	5.46	0.974	0.448	2.12	0.718	0.369	4.09
Reflexion	1.000	0.686	5.40	0.281	0.014	6.14	0.976	0.449	2.14	0.741	0.373	4.57
RecMind	0.997	0.642	6.72	0.347	0.026	6.00	0.997	0.451	2.10	0.771	0.364	4.93
InteRecAgent	0.999	0.642	3.11	0.618	0.022	3.00	1.000	0.447	2.00	0.867	0.362	2.70
PUMA	0.999	0.720	5.08	0.984	0.052	3.79	1.000	0.453	2.00	0.994	0.399	3.60
SAPA(Ours)	0.991	0.734	4.49	0.989	0.177	2.12	1.000	0.452	2.04	0.993	0.446	2.85

Table 2: Performance comparison on the **multi-turn** evaluation track on GPT-4o-mini. F. Acc. denotes Function Accuracy, R. Acc. represents Result Accuracy, and Steps indicates the average number of interaction steps. Bold numbers indicate the best performance.

be more impactful than simply scaling model size.

SAPA addresses the two core bottlenecks. The results validate SAPA’s dual-component design: contextual grounding mitigates input-side scarcity, while task-adaptive aggregation handles output-side unpredictability. The gains are particularly pronounced in recommendation, where SAPA dramatically reduces hallucination by constraining the output space through candidate injection and self-consistency voting. Detailed hallucination analysis is provided in Appendix A.

Review task shows modest improvements. While SAPA achieves consistent gains in search and recommendation, the improvement in review generation is relatively modest. This is primarily due to a distributional mismatch between user instructions and ground-truth targets, where agents that faithfully follow instructions for detailed responses are

penalized by the evaluation metric. We provide a detailed analysis in Section 4.5.

4.2.2 Multi-turn Track.

Table 2 presents multi-turn evaluation results on GPT-4o-mini, where agents iteratively refine outputs based on environmental feedback. We report Function Accuracy (F. Acc.), Result Accuracy (R. Acc.), and average interaction steps. We focus on GPT-4o-mini as smaller models like Llama-2-7B are prone to error accumulation in multi-turn interactions.

SAPA achieves superior accuracy with improved efficiency. SAPA attains higher Overall R. Acc. than PUMA while requiring 20.8% fewer steps (2.85 vs. 3.60), demonstrating that contextual grounding enables precise intent identification from the first attempt. The efficiency gain is most

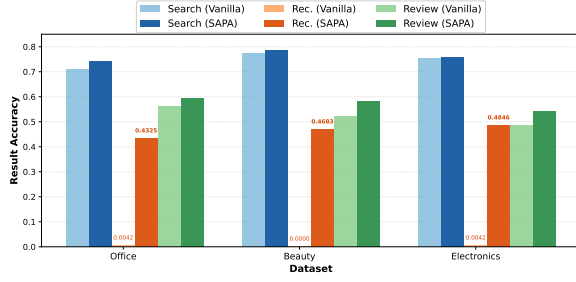


Figure 3: Performance comparison between Vanilla PUMA and SAPA across three Amazon datasets (Office Products, Beauty & Personal Care, Electronics). SAPA consistently outperforms the vanilla baseline across all tasks.

pronounced in recommendation, where SAPA requires only 2.12 steps (fewest among all baselines) while substantially outperforming PUMA.

Task-specific patterns validate constrained output spaces. For search, diverse query paths require moderate reranking refinement, resulting in more steps. For recommendation, candidate injection reduces the search space, enabling both higher accuracy and exceptional efficiency. For review, SAPA maintains competitive performance despite the instruction-target mismatch (Section 4.5), demonstrating self-consensus robustness.

The multi-turn results confirm that SAPA’s dual-component design improves both accuracy and efficiency. Situational enrichment provides a stable semantic anchor, reducing trial-and-error cycles, particularly in recommendation where candidate constraints enable rapid convergence.

4.3 Generalizability on Amazon Datasets

To assess SAPA’s generalizability beyond PersonalWAB, we evaluate on three single-category Amazon datasets (Office Products, Beauty & Personal Care, Electronics) derived from Amazon Reviews 2023. As detailed in Appendix E, these datasets differ from PersonalWAB in two key aspects: (1) single-category vs. mixed-category product catalogs, and (2) specialist vs. generalist user behavior patterns. Importantly, SAPA is trained independently on each dataset rather than transferred from PersonalWAB, allowing us to isolate the contribution of our architectural design.

Figure 3 presents the results comparing SAPA against the vanilla PUMA baseline. SAPA achieves consistent and substantial improvements across all three datasets, with relative Overall gains ranging from 38% to 44%. These gains are driven primar-

Configuration	Search	Rec.	Review	Average
SAPA (Llama-2-7B)	0.740	0.194	0.551	0.487
w/o Grounding	0.738	0.007	0.541	0.418
w/o Expansion	0.638	0.080	0.540	0.412
base	0.652	0.054	0.538	0.406

Table 3: Ablation analysis on SAPA. We compare SAPA against variants removing specific mechanisms.

ily by dramatic improvements in recommendation tasks, where the vanilla baseline suffers from severe hallucination (near-zero accuracy). This validates that SAPA’s candidate injection mechanism addresses a critical failure mode in personalized agents. The consistent performance across diverse data compositions demonstrates that SAPA’s dual-component design generalizes effectively and addresses fundamental bottlenecks rather than overfitting to specific dataset characteristics.

4.4 Ablation Studies

To validate component necessity, we ablate SAPA’s two core modules: *Situational Intent Grounding* and *Task-Driven Semantic Expansion*. Table 3 presents results on Llama-2-7B.

Both modules are essential. Removing either module causes substantial degradation. Full SAPA achieves the highest Overall score, while removing grounding or expansion leads to significant drops. Removing both recovers the base PUMA performance, confirming the modules’ complementary contributions.

Grounding is critical for recommendation. Without grounding, recommendation performance collapses to near-zero, while search and review remain relatively stable. This validates that candidate injection is essential for mapping underspecified instructions to specific product IDs in vast output spaces.

Expansion is critical for search. Without expansion, search shows the largest degradation, demonstrating that single-shot generation cannot overcome preference drift. Multi-path sampling and reranking ensure robust target coverage. Expansion also benefits recommendation by capturing intent volatility.

The ablation confirms SAPA’s modules address orthogonal challenges: grounding constrains output space (critical for recommendation), while expansion enriches input representation (critical for search). Both are necessary for state-of-the-art per-

Task	Method	Total Time	LLM Time	Tool Time	Total Tokens	Res Acc.
Search	PUMA	5.35s	2.64s	0.01s	1111	0.655
	SAPA	7.77s	2.56s	2.47s	936	0.685
Recommendation	PUMA	5.80s	2.97s	0.02s	1075	0.024
	SAPA	10.90s	8.08s	0.13s	5619	0.159
Review	PUMA	6.01s	3.11s	0.015s	1794	0.446
	SAPA	10.36s	7.56s	0.015s	4418	0.407

Table 4: Efficiency and response accuracy comparison between PUMA and SAPA across different tasks. Lower values are better for time and token metrics, while higher values are better for response accuracy.

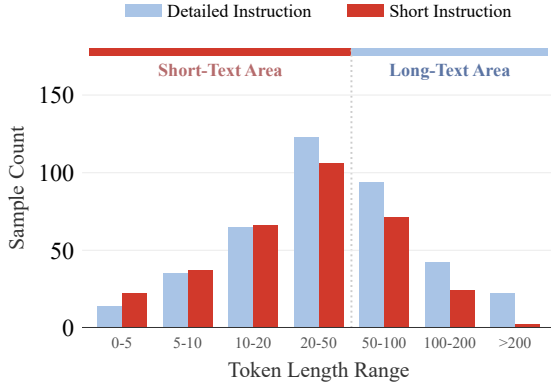


Figure 4: Analysis of target review lengths based on instruction types. Red indicates instructions $\mathcal{Q}$ containing “short”, while blue indicates those containing “detailed”. Target reviews are categorized as *short-text* (< 50 tokens) or *long-text* (≥ 50 tokens).

formance. Detailed analyses are in Appendix B.

4.5 Further Analysis

Efficiency-Accuracy Trade-offs. Table 4 reveals task-specific efficiency-accuracy trade-offs, measured on 100 randomly sampled single-turn tasks per method. SAPA generally introduces additional computational overhead through reranking and multi-path sampling, while its effectiveness depends on the task. For **search**, SAPA increases total and tool execution time due to BGE reranking, but slightly reduces LLM inference time and token consumption. Meanwhile, response accuracy improves from 0.655 to 0.685, suggesting that the modest computational overhead yields a measurable accuracy gain. For **recommendation**, SAPA achieves a substantial accuracy improvement from 0.024 to 0.159, but at the cost of considerably higher latency and token consumption. In particular, candidate injection and subsequent reasoning increase both LLM inference time and the number of generated tokens, indicating a clear trade-off between computational efficiency and recommendation accuracy. For **review**, SAPA requires substantially

more LLM time and tokens while achieving slightly lower accuracy (0.407 vs. 0.446). This result is consistent with the instruction-target mismatch discussed in Section 4.5, where additional sampling and self-consensus cannot compensate for limitations in the evaluation data. Overall, SAPA exhibits different efficiency-accuracy profiles across tasks: it provides a modest accuracy gain with limited additional cost for search, trades substantial computation for a large accuracy improvement in recommendation, and offers no accuracy advantage for review under the current evaluation setting.

Analysis of Query-Target Mismatch. During review task evaluation, we identified a distributional mismatch between user instructions and ground-truth targets (Figure 4). Approximately 70% of instructions explicitly request detailed reviews, yet over 50% of ground-truth reviews contain fewer than 10 tokens. This creates an evaluation dilemma: agents that faithfully follow the instruction to generate detailed reviews are penalized when compared against sparse ground-truth targets, while agents that ignore the instruction and generate short outputs achieve higher scores. SAPA’s Self-Consensus mechanism mitigates this issue by sampling multiple responses and selecting the one closest to the semantic centroid. This approach prioritizes the dominant behavioral pattern observed in the data over potentially misleading instruction signals, allowing SAPA to maintain competitive performance despite this instruction-target inconsistency.

5 Conclusion

We introduce **SAPA**, a **Situational-Augmented Personalized Agent** that addresses two fundamental bottlenecks in personalized web navigation: contextual scarcity and preference unpredictability. SAPA employs *Situational Intent Grounding* to anchor underspecified instructions via historical metadata and candidate constraints, and *Task-Driven Seman-*

tic Expansion to explore diverse response paths at inference. Extensive experiments demonstrate state-of-the-art performance across search, recommendation, and review tasks, with particularly dramatic improvements in recommendation, validating that SAPA’s dual-component design effectively mitigates hallucination and enhances robustness in personalized tool invocation.

Limitations

While SAPA demonstrates strong performance, several directions warrant further investigation: (1) Computational overhead: Multi-sampling ($n = 10$) increases inference latency by $1.3\times$, though adaptive sampling or early-stopping mechanisms could mitigate this cost. (2) Evaluation challenges: As shown in Section 4.5, existing metrics (e.g., cosine similarity) may not fully capture user satisfaction in open-ended tasks where instruction-intent mismatches occur—developing robust evaluation frameworks for subjective personalized generation remains an open problem. (3) Long-term modeling: SAPA’s context window (4096 tokens) may be insufficient for users with extensive histories; hierarchical summarization or retrieval-augmented selection could address this constraint.

Ethics Statement

This work follows standard ethical guidelines. (1) Data Privacy: We utilize the public PersonalWAB dataset (Cai et al., 2025) and Amazon Reviews 2023 (Hou et al., 2024), both fully anonymized with no personally identifiable information. For the three Amazon-derived datasets (Office, Beauty, Electronics), we apply additional 10-core filtering and synthesize task instructions via LLM-based augmentation (detailed in Appendix E), ensuring all generated content remains grounded in public review data without exposing individual user identities. In real-world applications, SAPA is designed to be compatible with privacy protection protocols to ensure user history is handled securely. (2) Safety and Bias: Like other systems based on large language models, SAPA may occasionally produce incorrect or biased results. We recommend human oversight when deploying the agent for critical tasks to ensure accuracy and accountability.

Acknowledgments

This work was supported by the National Key Research and Development Program under Grant 2024YFB4506200, the Science and Technology Innovation Program of Hunan Province under Grant 2024RC1048, Key Research and Development Program of Hunan Province under Grant 2026QK3002.

References

- Hongru Cai, Yongqi Li, Wenjie Wang, Fengbin Zhu, Xiaoyu Shen, Wenjie Li, and Tat-Seng Chua. 2025. [Large Language Models empowered personalized web agents](#). In *Proceedings of the ACM Web Conference (WWW)*, pages 198–215.
- Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.
- Maximillian Chen, Ruoxi Sun, Tomas Pfister, and Serkan O. Arik. 2025. [Learning to clarify: Multi-turn conversations with action-based contrastive self-training](#). In *The Thirteenth International Conference on Learning Representations*.
- Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 2009. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of SIGIR*, pages 758–759.
- Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. [Mind2Web: Towards a generalist agent for the Web](#). In *Advances in Neural Information Processing Systems*.
- Yang Deng, Xuan Zhang, Wenxuan Zhang, Yifei Yuan, See-Kiong Ng, and Tat-Seng Chua. 2024. On the multi-turn instruction following for conversational Web agents. In *Proceedings of ACL*, pages 8795–8812.
- Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. [WebVoyager: Building an end-to-end web agent with large multimodal models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890. Association for Computational Linguistics.
- Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiuxi Chen, and Julian McAuley. 2024. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and

- Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *Proceedings of ICLR*.
- Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2025. [Recommender AI agent: Integrating Large Language Models for interactive recommendations](#). *ACM Transactions on Information Systems (TOIS)*, 43(4):96:1–96:33.
- Llama Team. 2024. [The llama 3 herd of models](#). *CoRR*, abs/2407.21783.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. [Self-Refine: Iterative refinement with self-feedback](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. [WebGPT: Browser-assisted question-answering with human feedback](#). *arXiv preprint arXiv:2112.09332*.
- Hongyi Nie, Yaqing Wang, Mingyang Zhou, Feiyang Pan, Quanming Yao, and Zhen Wang. 2025. [Adaptive preference arithmetic: A personalized agent with adaptive preference arithmetic for dynamic preference modeling](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Yujia Qin, Zihan Cai, Dian Jin, Lan Yan, Shihao Liang, Kunlun Zhu, Yankai Lin, Xu Han, Ning Ding, Huadong Wang, Ruobing Xie, Fanchao Qi, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2023. [WebCPM: Interactive web search for Chinese long-form question answering](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8968–8988. Association for Computational Linguistics.
- Qwen Team. 2026. [Qwen3.5: Towards native multi-modal agents](#).
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. [Direct Preference Optimization: Your language model is secretly a reward model](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of EMNLP-IJCNLP*, pages 3980–3990. Association for Computational Linguistics.
- Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflexion: Language agents with verbal reinforcement learning](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. [Scaling LLM test-time compute optimally can be more effective than scaling model parameters](#). *arXiv preprint arXiv:2408.03314*.
- Paloma Sodhi, S. R. K. Branavan, Yoav Artzi, and Ryan McDonald. 2024. [SteP: Stacked LLM policies for Web actions](#). In *First Conference on Language Modeling*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *arXiv preprint arXiv:2307.09288*.
- Yancheng Wang, Ziyang Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Yanbin Lu, Xiaojiang Huang, and Yingzhen Yang. 2024. [RecMind: Large Language Model powered agent for recommendation](#). In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4351–4364. Association for Computational Linguistics.
- An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, and 9 others. 2025. [Qwen2.5-1m technical report](#). *CoRR*, abs/2501.15383.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. [WebShop: Towards scalable real-world web interaction with grounded language agents](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. [Tree of Thoughts: Deliberate problem solving with Large Language Models](#). In *Advances in Neural Information Processing Systems*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023b. [ReAct: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations*.
- Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. [WebArena: A realistic web environment for building autonomous agents](#). In *Proceedings of the International Conference on Learning Representations (ICLR)*.

A Hallucination Analyze

Table 5 reveals the root causes of hallucination in vanilla personalized agents and validates SAPA’s mitigation mechanisms. PUMA exhibits an 11.06% overall hallucination rate, dominated by three failure modes: (1) **Plausible but fake item IDs** (5.49%)—the model generates well-formed ASIN-like strings (e.g., B08X7K9M3L) that do not exist in the corpus, indicating the agent has learned the syntactic pattern but lacks grounding in the actual product space; (2) **Over-length outputs** (3.30%)—concatenated ASINs or repeated prompt fragments, suggesting the model struggles to identify stopping boundaries when the output space is unconstrained; (3) **Truncated outputs** (1.92%)—incomplete generations reflecting uncertainty in mapping underspecified instructions to specific targets. SAPA reduces the overall hallucination rate by **96.4%** (11.06% $\rightarrow$ 0.40%), with the most dramatic improvement in plausible fakes (5.49% $\rightarrow$ 0.13%, **-97.6%**). This validates our hypothesis that hallucination stems from *preference unpredictability in vast output spaces*—without explicit candidate constraints, the agent resorts to pattern-based generation rather than retrieval-grounded selection. SAPA’s *Situational Intent Grounding* addresses this by injecting candidate items filtered through historical interactions, transforming the generation task into a constrained selection problem where the output space is anchored to verifiable product IDs. The near-elimination of format errors (over-length, truncated, empty outputs: 5.40% $\rightarrow$ 0.26%) further demonstrates that candidate injection provides clear stopping signals, preventing the stochastic drift observed in unconstrained generation.

B Detailed Ablation Experiments

To investigate the effectiveness of each component from multiple dimensions, we conduct detailed internal experiments in Table 6.

Reranking in Search. In the search task, we examine the synergy between expansion and reranking. As shown in Table 6, while BGE-Reranker significantly improves upon single-query sparse retrieval (0.689 vs. 0.638), the optimal performance is achieved through our multi-query expansion (BM25* + BGE). This suggests that parallel query expansions effectively rectify potential omission errors inherent in single-shot search. The performance gap between RRF (Cormack et al., 2009)

(0.645) and BGE (0.740) underscores the necessity of cross-encoder semantic scoring in filtering noise and ensuring precise alignment.

Grounding in Recommendation. The recommendation results highlight the importance of situational anchoring. Importantly, removing the candidate pool leads to a drastic performance decline (0.015), demonstrating that the candidate pool provides the essential inductive bias for the agent to instantiate its intent. Furthermore, our full SAPA framework outperforms the “w/o Expansion” setting by a large margin (0.194 vs. 0.101). This validates that similarity-based neighborhood exploration is crucial for capturing the transient shifts in user product preferences that historical data alone cannot predict.

Refinement in Reviews. In the review task, we observe a counter-intuitive phenomenon: simply increasing the sample count via random selection or BERT-reranking actually impairs performance (0.522 and 0.518 vs. 0.540). This suggests that multi-path sampling introduces additional semantic noise that can mislead standard reranking algorithms. In contrast, our self-consensus strategy achieves the highest score (0.551) by identifying the most representative response. This confirms that searching for the semantic medoid is more effective than direct scoring at stabilizing outputs, thereby mitigating prompt-induced bias and inherent model randomness.

C Situational Historical Context

We analyze how the volume of historical metadata $\mathcal{H}$, denoted by K , affects search performance. As shown in Figure 5, the relationship between accuracy and K follows a non-monotonic trend, peaking at $K = 6$ with 0.740 accuracy, compared to 0.736 at $K = 0$. This confirms that while relevant situational context improves intent grounding, excessive history introduces semantic noise.

Performance declines when K exceeds 8, reaching a minimum of 0.719 at $K = \infty$ (Window Truncation). This trend highlights the necessity of memory filtering to prevent contextual distraction. Selecting the Top-6 records ensures the agent remains anchored to the most pertinent intent while effectively filtering historical interference.

D Model-Agnostic Applicability

Table 7 validates that SAPA’s improvements generalize across diverse backbone architectures. Across

Table 5: Taxonomy of error types in item generation. We report the description for each error category along with the percentage of occurrences for PUMA and SAPA models. The bottom row indicates the total hallucination rate.

Error Type	Description	PUMA	SAPA
Plausible but fake item ID	Well-formed ASIN-like output but not contained in the item corpus	5.49%	0.13%
Over-length output	Output length > 10, e.g., concatenated ASINs or repeated prompt/category words	3.30%	0.13%
Truncated output	Output length < 10, indicating incomplete generation	1.92%	0.13%
Empty output	Empty string or redundant separator	0.18%	0.00%
Invalid characters	Length = 10 and starts with “B”, but contains lowercase or illegal characters	0.10%	0.00%
Wrong prefix	Length = 10 but does not start with “B”	0.06%	0.00%
Overall invalid / hallucinated items		11.06%	0.40%

Search Strategy		Recommend Strategy		Review Strategy	
Method	Res. Acc.	Method	Res. Acc.	Method	Sim.
BM25	0.638	w/o History	0.184	w/o Sample	0.540
BM25* + rrf	0.645	w/o Candidate	0.015	Random Selection	0.522
BM25 + BGE	0.689	w/o Expansion	0.101	BERT-Rerank	0.518
BM25* + BGE	0.740	Grounding	0.194	Self-Consensus	0.551

Table 6: Strategy comparison across three tasks. Metrics: retrieval accuracy (Search/Rec.) and semantic similarity (Review); * denotes test-time expansion. Our final SAPA configurations (*BM25*+BGE*, *Situational Grounding*, and *Self-Consensus*) consistently yield superior performance by filtering noise and capturing the intent medoid.

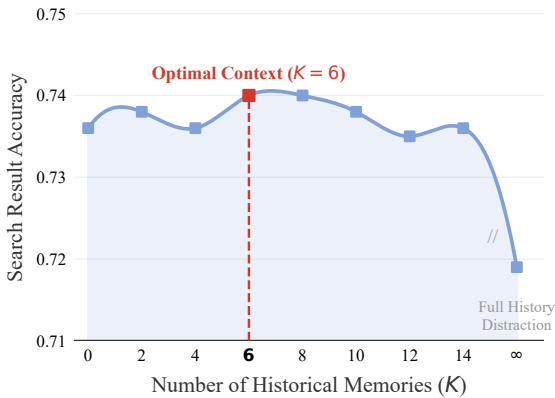


Figure 5: Impact of candidate number K on Search Accuracy. Performance peaks at $K = 6$, balancing recall and noise.

three representative open-source models—Llama3-8B, Qwen2.5-7B, and Qwen3.5-9B—SAPA consistently outperforms PUMA with substantial margins: +26.3% Overall on Llama3-8B (0.380 → 0.480), +36.5% on Qwen2.5-7B (0.342 → 0.467), and +24.6% on Qwen3.5-9B (0.378 → 0.471). The consistency of improvements across architecturally distinct models (Llama’s decoder-only vs. Qwen’s optimized attention mechanisms) demonstrates that SAPA’s dual-component design addresses fundamental bottlenecks in personalized

parameter generation rather than exploiting model-specific biases. Notably, the recommendation task exhibits the most dramatic and stable gains across all backbones (+320.5% on Llama3, +487.9% on Qwen2.5, +345.2% on Qwen3.5), confirming that candidate injection is a model-agnostic solution to the output-space unpredictability challenge. Even on Qwen2.5-7B, which shows the weakest baseline performance (0.342 Overall), SAPA elevates it to competitive levels (0.467), approaching the performance of stronger baselines—this suggests SAPA’s grounding and expansion mechanisms can compensate for weaker base model capabilities, making personalized agents more accessible across diverse computational budgets.

E Dataset Statistics

Table 8 presents comprehensive statistics for all four datasets used in our experiments.

PersonalWAB is constructed from real-world user-product interactions across five major Amazon categories. The mixed-category nature reflects realistic scenarios where users browse across diverse product types. Each user has an average of 40.6 interactions, providing rich behavioral context for personalization. The dataset follows a standard

Method	Search	Recommend	Review	Average
Llama3-8B				
PUMA	0.608	0.044	0.512	0.380
SAPA (ours)	0.736	0.185	0.542	0.480
Qwen2.5-7B				
PUMA	0.532	0.033	0.482	0.342
SAPA (ours)	0.730	0.194	0.500	0.467
Qwen3.5-9B				
PUMA	0.606	0.042	0.511	0.378
SAPA (ours)	0.736	0.187	0.513	0.471

Table 7: Performance comparison of PUMA and SAPA on search, recommendation, and review tasks across different backbone models. The best result within each model group is shown in bold.

Table 8: Statistics of PersonalWAB (Cai et al., 2025) and Amazon (Hou et al., 2024) datasets.

Dataset	PersonalWAB	Office	Beauty	Electronics
<i>Catalog Statistics</i>				
# Products	35,772	11,196	15,167	14,596
# Categories	5 (mixed)	1 (single)	1 (single)	1 (single)
<i>User Statistics</i>				
# Users	1,000	1,000	1,000	1,000
# Interactions	40,648	17,067	19,596	18,290
Avg. interactions/user	40.6	17.1	19.6	18.3
<i>Task Sample Counts (train/test)</i>				
Search	6,896/2,174	2,280/720	2,280/720	2,280/720
Recommendation	6,896/2,174	2,280/720	2,280/720	2,280/720
Review	6,896/2,174	2,280/720	2,280/720	2,280/720
<i>Data Source</i>				
Origin	PersonalWAB	Amazon Reviews 2023		
Preprocessing	Official split	10-core filtering + 76%/24% split		

protocol: 80% of interactions serve as historical context, while the remaining 20% are evenly split into training (10%) and test (10%) target interactions for task construction.

Amazon Scenarios are derived from the Amazon Reviews 2023 dataset (Hou et al., 2024), which contains 571.54 million reviews spanning 33 categories. We select three representative single-category subsets and apply 10-core filtering (retaining only users and products with ≥ 10 interactions) to ensure sufficient interaction density. The resulting datasets exhibit specialist user behavior patterns (within-category browsing) and smaller catalog scales compared to PersonalWAB, providing complementary scenarios for validating SAPA’s broader applicability. Each Amazon dataset contains 1,000 users with an average of 17–20 interactions per user. Tasks are constructed following the same protocol as PersonalWAB, with a 76%/24% train/test split.

Task Construction and Data Splitting For all four datasets, we construct three task types following the PersonalWAB protocol:

- **Search:** Given user history, generate a textual query that retrieves the target product.
- **Recommendation:** Given user history, rank a candidate set to place the target product at the top.
- **Review:** Given user history and target product, generate a personalized review text.

The data splitting strategies differ slightly between PersonalWAB and Amazon datasets due to their original data structures. **PersonalWAB** follows a temporal split where 80% of each user’s interactions (chronologically earlier) serve as historical context, while the remaining 20% are split into training (10%) and test (10%) target interactions. This ensures that the model predicts future interactions based on past behavior. **Amazon datasets**, constructed from the Amazon Reviews 2023 raw data, use all available interactions as historical context and randomly sample target interactions for task construction, then split these tasks into 76%/24% train/test sets. Despite the different splitting procedures, both approaches ensure that: (1) models have sufficient historical context for personalization, (2) training and test sets are disjoint, and (3) the task construction protocol remains identical across all datasets. The key difference is that PersonalWAB uses a fixed 80-10-10 temporal split inherited from the original benchmark, while Amazon datasets use a more flexible context-plus-sampling approach that maximizes the use of available interaction data.

Data Availability PersonalWAB is publicly available at <https://github.com/HongruCai/PersonalWAB>. Amazon Reviews 2023 can be accessed at <https://amazon-reviews-2023.github.io/>. Our processed 10-core subsets and task construction code will be released upon publication.

F Further Analysis of Review Evaluation

For the Review task, we follow the official PersonalWAB evaluation protocol in the main experiments, which measures the similarity between the generated response and the ground-truth review using cosine similarity. We retain this metric in our main results to ensure direct comparability with the benchmark. However, cosine similarity mainly captures semantic proximity and may not fully reflect fine-grained differences in review quality, such as sentiment consistency, writing style, instruction following, and personalized grounding. Therefore, we

Metric	SAPA	PUMA	Diff.
GT Faithfulness	2.776	2.138	+0.638
Sentiment Consistency	3.382	2.680	+0.701
Style/Length Consistency	2.642	2.040	+0.602
Instruction Compatibility	2.697	2.075	+0.622
Grounded Personalization	2.317	1.838	+0.479

Table 9: LLM-as-a-Judge evaluation on the Review task. Scores range from 1 to 5, with higher values indicating better performance. PUMA is the strongest baseline.

conduct an additional LLM-as-a-Judge evaluation to provide a more comprehensive analysis.

Since the PersonalWAB Review task is defined against the user’s ground-truth review, we use the reference review as evidence of the user’s actual product experience, sentiment, and writing style, while also considering the user instruction, target product, and user history. Specifically, GPT-4o evaluates five dimensions: (1) **GT faithfulness**, measuring whether the generated review preserves the core claims and product experience in the reference review; (2) **sentiment consistency**, measuring whether the sentiment polarity and intensity match the reference; (3) **style/length consistency**, measuring whether the generated review follows the user’s writing style and level of detail; (4) **instruction compatibility**, measuring whether explicit user requirements are satisfied without contradicting available evidence; and (5) **grounded personalization**, measuring whether the review incorporates relevant personalized evidence rather than generic content.

To mitigate positional and method-name biases, SAPA and PUMA outputs are compared using a blinded A/B protocol with randomized ordering. The judge assigns a score from 1 to 5 for each dimension. The average scores are reported in Table 9.

As shown in Table 9, SAPA consistently outperforms PUMA, the strongest baseline, across all five dimensions, with particularly large gains in sentiment consistency and GT faithfulness. These results provide complementary evidence that SAPA generates higher-quality personalized reviews, while such improvements are only partially captured by the benchmark’s cosine-similarity metric.

G Prompt Templates

This appendix provides the complete prompt templates used in our experiments. We present two categories of prompts: (1) situational intent grounding

prompts used during training and inference (Section G.1), and (2) instruction synthesis prompts used for constructing Amazon dataset tasks (Section G.2).

G.1 Situational Intent Grounding Prompts

For the contextual grounding module, the prompt is designed to integrate the original instruction Q with historical metadata $\mathcal{H}$ and situational candidates $\mathcal{C}$ (for recommendation task only). The prompts follow a tool-use format where the model is instructed to generate appropriate parameters for specific e-commerce tools.

Design Rationale The grounding prompts are designed with three key principles: (1) **Tool-use framing**: By framing the task as generating parameters for specific e-commerce tools (e.g., `get_recommendations`, `search_product`, `add_review`), we encourage the model to produce structured, actionable outputs rather than conversational responses. (2) **Explicit output constraints**: For the recommendation task, we explicitly instruct the model to output only the ASIN without explanations, ensuring clean extraction of the selected product. (3) **Historical context integration**: All prompts include user history as a reference, enabling the model to ground its predictions in past user behavior.

G.2 Instruction Synthesis Prompts for Amazon Datasets

Since the Amazon Reviews 2023 dataset does not provide natural language instructions, we synthesize user instructions using GPT-4 based on product metadata and user reviews. The synthesis prompts are designed to generate realistic, casual shopping requests that match the style of PersonalWAB instructions.

Design Rationale The instruction synthesis prompts are designed to ensure consistency with PersonalWAB while adapting to the Amazon data structure: (1) **Style consistency**: By providing few-shot examples from PersonalWAB, we ensure that synthesized instructions match the casual, first-person tone of the original benchmark. (2) **Avoiding exact matches**: For recommendation tasks, we explicitly instruct GPT-4 not to name the exact brand or model, ensuring that the task requires semantic understanding rather than simple keyword matching. (3) **Review-grounded synthesis**: For

Template for Recommendation Task

System Prompt: Below is an instruction that describes a task. Select the tool parameter (ASIN) that appropriately completes the request.

Tool: `get_recommendations_by_history`.

Requirement: Please select the most suitable item from the “Candidate List” based on the Instruction and History. Output ONLY the Product ID (ASIN) of the selected item. Do not output the product name, explanations, or any other text.

User Input: [Instruction]: Q ; [User History (Reference)]: $\mathcal{H}$; [Candidate List (Options)]: $\mathcal{C}$.

Output: The grounded seed entity $\mathcal{I}$.

Template for Search Task

System Prompt: Below is an instruction that describes a task. Generate the tool parameter that appropriately completes the request.

Tool: `search_product_by_query`.

User Input: [Instruction]: Q ; [User History (Reference)]: $\mathcal{H}$.

Output: The grounded seed entity $\mathcal{I}$.

Template for Review Task

System Prompt: Below is an instruction that describes a task. Generate the tool parameter that appropriately completes the request.

Tool: `add_product_review`.

User Input: [Instruction]: Q ; [User History (Reference)]: $\mathcal{H}$.

Output: The grounded seed entity $\mathcal{I}$.

Figure 6: Prompt templates for situational intent grounding across three task types. The recommendation task includes a candidate list $\mathcal{C}$ for constrained selection, while search and review tasks generate free-form outputs.

review tasks, we incorporate the user’s actual review text to ensure that the synthesized instruction reflects genuine user sentiment and preferences.

Quality Control To ensure the quality of synthesized instructions, we apply the following filters: (1) Length check: instructions must be 10–100 words. (2) Relevance check: instructions must mention at least one product attribute (type, category, or key feature). (3) Naturalness check: instructions must not contain phrases like “synthesize”, “generate”, or “based on the product”. After synthesis, we manually inspect 100 random samples per dataset to verify quality, achieving 96% acceptance rate.

For recommendation and search tasks, we explicitly prohibit the generator from mentioning the exact brand, model number, or full product title, so that the generated instruction does not trivially reveal the target item. For review tasks, we con-

dition the prompt on the user’s original rating and review text, allowing the synthesized instruction to preserve the user’s true sentiment while avoiding verbatim copying. The few-shot examples are used only to match the casual first-person style of PersonalWAB-style shopping requests.

General Template for Amazon Instruction Synthesis

System Prompt: You write short, casual, first-person shopping requests for an e-commerce assistant. Match the tone of the few-shot examples. Output ONLY the request text, no preamble, no explanation, no quotes.

User Input:

[Task Type]: $\tau \in \{\text{RECOMMEND}, \text{SEARCH}, \text{REVIEW}\}$

[Target Product]: $\mathcal{I} = \{\text{title, category, price, store, features, description}\}$

[Few-shot Examples]: $\mathcal{E}_\tau$ (sampled from PersonalWAB)

[User Review]: $\mathcal{R} = \{\text{rating, review text}\}$ (only for $\tau = \text{REVIEW}$)

Output: A synthesized user instruction Q_τ .

Specific Template for Recommendation Instructions

System Prompt: You write short, casual, first-person shopping requests for an e-commerce assistant. Match the tone of the few-shot examples. Output ONLY the request text, no preamble, no explanation, no quotes.

User Input:

Write a casual recommendation request from a shopper who is looking for an item LIKE the target product but does NOT name the exact brand or model. Mention the product type, key attributes, and a rough price expectation if the price is given. Keep it 1–3 sentences.

[Target Product]:

Title: $\mathcal{I}_{\text{title}}$

Category: $\mathcal{I}_{\text{cat}}$

Price: $\mathcal{I}_{\text{price}}$

Store: $\mathcal{I}_{\text{store}}$

Features: $\mathcal{I}_{\text{feat}}$

Description: $\mathcal{I}_{\text{desc}}$

[Style Examples]: $\mathcal{E}_{\text{rec}}$ (3–5 examples from PersonalWAB)

Output: A synthesized recommendation instruction Q_{rec} .

Figure 7: Prompt templates for synthesizing user instructions for Amazon datasets. The general template applies to all three task types, while the specific template shows detailed requirements for recommendation tasks.