Large Language Models Empowered Personalized Web Agents

Hongru Cai
henry.hongrucai@gmail.com
National University of Singapore
Singapore

Yongqi Li*
liyongqi0@gmail.com
The Hong Kong Polytechnic University
Hong Kong SAR, China

Wenjie Wang*
wenjiewang96@gmail.com
University of Science and Technology of China
Hefei, China

Fengbin Zhu
zhfengbin@gmail.com
National University of Singapore
Singapore

Xiaoyu Shen
xyshen@eitech.edu.cn
Eastern Institute of Technology, Ningbo
Ningbo, China

Wenjie Li
cswjli@comp.polyu.edu.hk
The Hong Kong Polytechnic University
Hong Kong SAR, China

Tat-Seng Chua
dcscts@nus.edu.sg
National University of Singapore
Singapore

Abstract

Web agents have emerged as a promising direction to automate Web task completion based on user instructions, significantly enhancing user experience. Recently, Web agents have evolved from traditional agents to Large Language Models (LLMs)-based Web agents. Despite their success, existing LLM-based Web agents overlook the importance of personalized data (*e.g.,* user profiles and historical Web behaviors) in assisting the understanding of users' personalized instructions and executing customized actions.

To overcome the limitation, we first formulate the task of LLM-empowered personalized Web agents, which integrate personalized data and user instructions to personalize instruction comprehension and action execution. To address the absence of a comprehensive evaluation benchmark, we construct a **Personal**ized **W**eb **A**gent **B**enchmark (PersonalWAB), featuring user instructions, personalized user data, Web functions, and two evaluation paradigms across three personalized Web tasks. Moreover, we propose a **P**ersonalized **U**ser **M**emory-enhanced **A**lignment (PUMA) framework to adapt LLMs to the personalized Web agent task. PUMA utilizes a memory bank with a task-specific retrieval strategy to filter relevant historical Web behaviors. Based on the behaviors, PUMA then aligns LLMs for personalized action execution through fine-tuning and direct preference optimization. Extensive experiments validate the superiority of PUMA over existing Web agents on PersonalWAB. We release code and data at PersonalWAB github repository.

CCS Concepts

• **Information systems** → **Web applications**; **Personalization**.

*Corresponding authors.

WWW '25, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1274-6/25/04
https://doi.org/10.1145/3696410.3714842

Keywords

Personalized Web Agents, Personalization, Large Language Model

ACM Reference Format:
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 2025 (WWW '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3696410.3714842

1 Introduction

The World Wide Web has evolved into a fundamental infrastructure in the information age, with diverse Web services integrated into users' daily lives, including information retrieval, online shopping, and social engagement. However, the unprecedented scale and complexity of modern Web services also present new challenges. Users, particularly the elderly groups, are overwhelmed with vast amounts of unstructured data and intricate interactions, complicating task completion on the Web [54]. To alleviate the burden of complex Web operations, Web agents have emerged as a promising solution [41] to bridge users and Web services as shown in Figure 1(a). Based on user instructions, Web agents autonomously interact with the Web to complete tasks such as information retrieval and online shopping [55], offering a convenient way to enhance efficiency and intelligence with extensive Web services.

The evolution of Web agents has undergone a significant transition from traditional agents to those powered by Large Language Models (LLMs). Traditional agents typically optimize Web navigation tasks by reinforcement learning techniques [27, 41, 55], while their context understanding and reasoning capabilities are limited, failing to generalize to complex and out-of-distribution scenarios [6]. In recent years, LLMs have demonstrated extensive world knowledge along with strong understanding, planning, and reasoning capabilities, making LLM-based Web agents a rapidly evolving direction [10, 33]. Related research has leveraged techniques such as in-context learning [20, 45, 59, 60], fine-tuning [6, 13], and reinforcement learning [34] to enhance the instruction-following

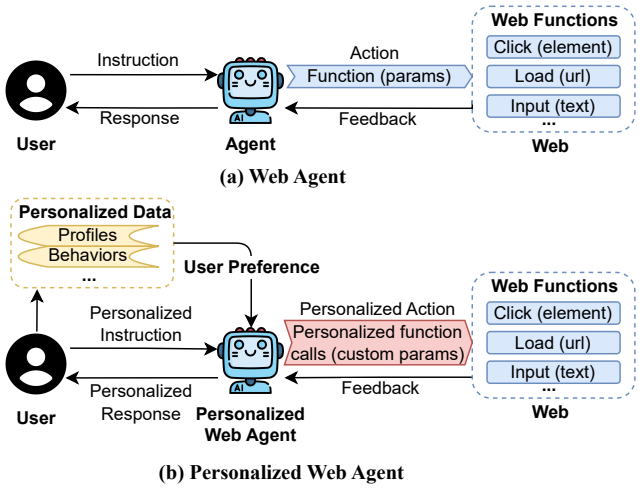


Figure 1: Comparison between traditional Web agents (a) and personalized Web agents (b). Personalized Web agents adopt personalized data to infer implicit user preferences, assisting in understanding user instructions and performing personalized actions, leading to more intelligent Web services.

capabilities of LLMs in various Web agent tasks. Notably, in addition to single-turn user instructions, some studies have explored utilizing the powerful interactive capabilities of LLMs to enable multi-turn interactions with users, facilitating conversational Web navigation and task execution [7, 29, 46].

Despite the significant success of LLM-based Web agents, they overlook the critical role of personalized data in enhancing user experience, as illustrated in Figure 1. Personalized data, such as user profiles and historical Web behaviors, reveals implicit user preference, which can facilitate the understanding of user instructions and enable personalized action execution. Specifically, 1) personalized data can supplement user context for ***personalized instruction comprehension***. For example, when users search for a product, their behavior history may reveal implicit preferences for product attributes (*e.g.,* price) that are not explicitly stated in user instructions. Besides, 2) personalized data enables ***personalized action execution***, where actions can be formulated as various Web function calls from Web services [22, 55, 58, 62]. In this work, we abstract the Web services from various websites (*e.g.,* , Amazon shopping website) as a diverse set of *"Web functions*[1]*"*; for example, the "search" action can be executed by passing a textual query as function parameters to the search API of a website. Users have varying habits and preferences for Web services, leading to personalized function calls with customized parameters.

In this light, we formulate the task of **LLM-empowered personalized Web agents**, which integrate personalized user data for personalized instruction comprehension and action execution, aiming to align with explicit user instructions and implicit user preferences derived from personalized data. Formally, given user instructions alongside personalized user data (*e.g.,* profiles and historical Web behaviors), LLMs must infer personalized user requirements and preferences to determine which Web function to call and formulate the corresponding function parameters. Subsequently, the results of these function calls are returned to users. However, to advance this task, the primary challenge is the lack of a comprehensive benchmark for training and evaluation.

[1]"Web APIs" and "Web tools" are also used to convey similar meanings in agents. For convenience, we unify these terms as "Web functions" below.

Table 1: Comparison between existing benchmarks and PersonalWAB from three key aspects: interaction type with users, Web environment type for agent interactions, and utilization of personalized data.

Benchmark	Interaction Type	Environment Type	Personalization
MiniWoB++ [27]	Single-turn	Mobile Web UI	×
RUSS [54]	Multi-turn	Web UI	×
META-GUI [46]	Multi-turn	Mobile apps	×
WebShop [55]	Single-turn	Shopping Web UI	×
Mind2Web [6]	Single-turn	Web UI	×
WebArena [62]	Single-turn	Web UI	×
VWA [22]	Single-turn	Web UI	×
WebVoyager [14]	Single-turn	Web UI	×
WorkArena [9]	Multi-turn	Web UI	×
WebLINX [29]	Multi-turn	Web UI	×
MT-Mind2Web [7]	Multi-turn	Web UI	×
MMInA [58]	Single-turn	Web UI	×
Turking [53]	Single-turn	Web UI	×
ChatShop [4]	Multi-turn	Web function	×
PersonalWAB	Single-turn & Multi-turn	Web function	✓

To bridge this gap, we construct the first **Personal**ized **W**eb **A**gent **B**enchmark (PersonalWAB). PersonalWAB focuses on three representative tasks for personalized Web agents: personalized search, recommendation, and review generation on Web platforms, which require LLMs to infer user preferences for task completion. Specifically, PersonalWAB is constructed by the following steps:

1) *Personalized Data Construction*: PersonalWAB adopts Amazon review dataset [15] to construct 1,000 diverse users with simulated profiles and real Web behaviors (*e.g.,* purchase and rating).
2) *User Instruction Creation*: For the three tasks, PersonalWAB utilizes users' genuinely liked items as the ground truth for search and recommendation, and real reviews as the ground truth for review generation. PersonalWAB then uses ground truth items and reviews to synthesize corresponding user instructions.
3) *Web Environment Implementation*: To interact with the Web environment, PersonalWAB develops a series of Web functions for the three tasks.
4) *Evaluation*: PersonalWAB utilizes the ground truth in Step 2 to assess the three tasks. Notably, it not only supports single-turn evaluation but also develops an LLM-based user simulator for real-time multi-turn interaction and evaluation with users.

Extensive analysis in Section 4 validates that PersonalWAB offers a set of users with diverse profiles and behaviors, and the simulated user profiles closely align with the actual behaviors by empirical evaluation. The comparison with existing benchmarks is in Table 1.

To enable LLM-empowered personalized Web agents, we propose a **P**ersonalized **U**ser **M**emory-enhanced **A**lignment (PUMA) framework. PUMA stores users' long-term Web behaviors into a memory bank and utilizes a task-specific retrieval strategy to filter out irrelevant information, focusing only on behaviors and features relevant to the current instruction. Given the retrieved behaviors and features, PUMA then combines them with user instructions

to call appropriate Web functions and generate optimal function parameters to enhance the returned results. However, the large parameter space challenges LLMs in producing high-reward parameters. To address this, PUMA designs several heuristic strategies to construct pseudo-label parameters for supervised fine-tuning (SFT) [35], enabling LLMs to generate reasonable function parameters. PUMA then uses Direct Preference Optimization (DPO) [36] to sample multiple function parameters for pair-wise optimization, better aligning with personalized user preferences. Experimental results demonstrate that PUMA significantly outperforms existing Web agents in single-turn and multi-turn personalized Web tasks, showcasing the potential of personalized Web agents to deliver more intelligent, customized, and user-centered Web services.

The key contributions in this work are as follows:

- We are the first to formulate the task of LLM-empowered personalized Web agents, which incorporate personalized user data to achieve personalized instruction understanding and action execution, bridging users with customized Web services.
- We construct the first benchmark for LLM-empowered personalized Web agents, featuring a diverse set of users with varying profiles and behaviors, the instructions across three tasks, callable Web functions, and two evaluation paradigms.
- We propose PUMA, a novel personalized alignment framework with a user memory bank and optimization strategies to align LLMs with the task of personalized Web agents.
- We conduct extensive experiments on PersonalWAB, showing that PUMA consistently surpasses existing Web agents, aligning better with personalized user instructions and preferences.

2 Related Work

• **Web agents.** Web agents are designed to automate a variety of complex Web-based tasks. Some studies focus on directly responding to users' instructions in a single turn. MiniWoB++ [27] established a platform of website widgets where agents can complete online tasks through keyboard and mouse. Webshop [55] introduced a simulated e-commerce environment with human-written task instructions. Recent studies investigate automating Web tasks under more practical and complex settings, including multi-domain [6], multi-hop [58], real-time interactions with Web [62], and visual UI understanding [14, 22]. Numerous efforts have been made to solve these problems, including fine-tuning [11, 13, 32] and prompting LLMs [42, 56, 59]. 2) Another research direction involves integrating user interactions into the agent's execution process. META-GUI [46] introduced a dataset focused on automating actions in mobile apps following conversational user instructions. RUSS [54] designed a dataset to boost dialogue-centric Web navigation. Recent works also focus on conversational Web navigation [7, 29] and interactive information-seeking problems [4].

Despite advancements, prior research overlooks the dimension of personalization in Web agents. A recent study simulates users with distinct roles, permissions, and interaction histories [62], but these roles are predefined per platform and do not require understanding user preferences nor demand the agent to adjust the execution strategy according to user preferences. In this work, we first focus on LLM-empowered personalized Web agents and propose a novel framework along with a benchmark for the optimization and evaluation of LLM-empowered personalized Web agents.

• **Personalized LLMs.** Personalized LLMs are designed to handle user personas (*e.g.,* , background information or historical behaviors) to meet individualized needs, adapting to distinct users [48]. Research in this field falls into two main categories: personalized content generation and user-facing applications. 1) Personalized content generation focuses on the core challenges of generating personalized content. They have used openly available user data on Reddit [50], Facebook, Twitter [43], and other blogging websites [21] to pre-train LLMs. Key tasks include stance classification, demographic inference [44], and personalized sentiment prediction [31]. Benchmarks like LaMP [39] and LongLaMP [23] further provide datasets for evaluating personalized text classification and content generation. 2) Another research direction is the practical applications in real-world scenarios, starting with personalized dialogue systems. Studies have built dialogue datasets by promoting crowd-workers to author dialogues based on specific personas [57], and by extracting user attributes from Reddit [30] and Weibo [61]. Apollonion [5] dynamically updates user profiles for personalized responses. Additionally, memory mechanisms [24, 28, 52] help models recall past conversations and important events. Personalized LLMs are also applied in healthcare [1, 18], education [8, 40], and robotics [51] to enhance services.

However, previous studies have not explored personalized function calls tailored to user-specific needs. Our work bridges this gap by emphasizing adapting agents' actions to different users by utilizing personalized user data and enabling a comprehensive assessment of agents' ability to complete several personalized tasks in Web environments.

3 Task and Benchmark

In this section, we formulate the task of LLM-empowered Web agents in Section 3.1, detail the construction of PersonalWAB in Section 3.2, and present the evaluation paradigms in Section 3.3.

3.1 Task Formulation

LLM-empowered personalized Web agents act as intermediaries between users and Web services, and we formulate the following elements in this task:

- User. Each user $u \in \mathcal{U}$ has a distinct profile P_u and the historical Web behaviors H_u. The profile P_u includes static attributes such as demographics, and H_u records the user's past Web behaviors, represented as a time-ordered sequence, $\{h_u^1, h_u^2, \ldots, h_u^N\}$. Each h_u^i denotes one Web behavior, such as a purchase or a review.
- Instruction. The user's instruction i_u is a natural language sentence that expresses their needs and requirements.
- Web environment. It is abstracted as a series of Web functions, denoted by $\mathcal{T}$. Each function $f \in \mathcal{T}$ can be invoked with an input parameter p, returning the corresponding result O_{f_p}. Notably, different input parameters will yield different function results.

Given the user instruction i_u and the personalized data P_u and H_u, LLM-empowered personalized Web agents aim to select the appreciate Web function f and determine the optimal parameter p to invoke personalized results O_{f_p} from the Web environment.

Figure 2: Overall pipeline for constructing PersonalWAB benchmark with a real example for recommendation instruction.

3.2 Benchmark Construction

It is challenging to gather a set of users to collect the real data on Web agent applications. Therefore, we chose to develop the benchmark using existing datasets of users' Web behaviors. Specifically, we selected the Amazon Review [15] dataset as the foundation for our benchmark, as it provides a large-scale collection of users' Web behaviors, including purchase and product rating across various categories of products. The overall pipeline for constructing PersonalWAB is illustrated in Figure 2, which consists of three steps: *personalized data construction*, *user instruction creation*, and *Web environment implementation*.

3.2.1 Personalized Data Construction. This section consists of user sampling and user profile generation steps.

• **User sampling.** We randomly selected 1,000 users from the Amazon Review across five distinct product categories: *Electronics*, *Home and Kitchen*, *Grocery and Gourmet Food*, *Clothing, Shoes, and Jewelry*, and *Health and Household*. For each user, we collected all their interactions across the five categories, each containing detailed purchased product information (such as product title, price, rating, and store) and user evaluations (including ratings, titles, and comments).

To simulate realistic user behavior, we first arranged all user interactions chronologically and divided them based on timestamps into three segments: 80% as historical data, 10% for the training set, and the final 10% for the test set.

• **User profile generation.** We generated unique profiles for each of the 1,000 users based on their entire history of behaviors, using the language model to infer and summarize their potential profiles (see prompt in Figure 10). Each user profile is structured to reflect the following key dimensions (see details in Figure11):

- Basic information. This includes fundamental personal attributes such as *gender*, *age*, and *occupation*, inferred from the user's product categories and purchasing behaviors.
- Shopping preferences. This dimension captures the user's purchasing tendencies, including 1) *price sensitivity* (whether the user gravitates towards budget, mid-range, or premium products), 2) *shopping interests* (the types of products the user is most frequently drawn to), and 3) *brand preferences* (the brands most commonly referenced in the user's purchase history).
- Behavioral tendencies. We summarize the characteristics of each user's shopping behavior using LLM from the following aspects. 1) *Diversity preference* indicates whether the user favors trying new products or sticking with familiar ones; 2) *Interaction complexity* describes whether the user prefers concise or detailed interactions based on their review patterns; 3)*Tone and style* capture the emotional tone and expressive style of the user's reviews, which may affect how they engage with systems. 4) *Item reference* examines how often the user refers to specific products or brands in their reviews; and 5) *focus aspects* highlight which product features (e.g., price, average rating, brand) the user tends to prioritize in their reviews.

The user profiles will support the following personalized instruction generation (§ 3.2.2) and multi-turn evaluation (§ 3.3).

3.2.2 User Instruction Creation. As previously mentioned, organizing thousands of users to collect their real instructions poses significant challenges. To address this, we prompt the LLM to generate personalized instructions based on each user's profile and real Web behaviors. These instructions encompass three task scenarios: search, recommendation, and review.

- Search instructions: We provide the language model with a detailed user profile and product information, including key attributes like brand, category, and features, to generate instructions for searching products. The prompt is detailed in Figure 12. Depending on the profile, the generated search instructions vary in length, tone, level of detail regarding the product, and the specific product aspects mentioned.
- Recommendation instructions: The recommendation instruction requests generated tend to be shorter, more general, and less specific, leaving room for broader exploration. We prompt the LLM to generate recommendation tasks with the prompt (see Figure 13), user profile, and the user's integrated products.
- Review instructions: The LLM receives both the user profile, target product information, and actual review text to generate user instructions for writing the review with users' personalized requirements. The prompt details are shown in Figure 14.

3.2.3 Web Environment Implementation. We choose to abstract and simplify the Web environment as a series of Web functions [4] rather than Web GUIs [53], as we believe GUIs are primarily user-friendly interfaces for humans and not essential for agents. The following web functions have been developed to help the agent complete users' instructions:

- **search_product_by_query.** This function takes a textual query as input and returns detailed information on the 10 most similar products based on the query. We facilitate this function using

BM25 with Pyserini [26] to enable fast retrieval from a database of all products.

- **get_recommendations_by_history.** This function accepts a sequence of product IDs and returns 10 recommended products. To implement this, we trained the SASRec model [19] on our conducted benchmark, with cold-start products removed.
- **add_product_review.** Designed to simplify the process of adding a product review, the only parameter this function requires is the review text. We assume the review is posted on the website once this function is successfully invoked.
- **respond.** This function allows the agent to engage in dialogue with the user, enabling clarification or gathering of additional information.
- **stop.** The stop function signals the termination of the current task. When invoked, it indicates that the agent decided to end the task, and no further actions are required.

3.3 Evaluation

To thoroughly evaluate the capabilities of Web agents, we established two distinct evaluation tracks: the single-turn track and the multi-turn track.

• **Single-turn track.** In this track, the agent is given only one opportunity to execute the user's instruction. The Web agent is expected to invoke the appropriate Web functions and deliver accurate results by configuring these functions with optimal parameters. Therefore, we define two metrics as follows:

- **Function accuracy** (function acc): This metric assesses the agent's ability to select the correct function and provide parameters in the correct format. If the agent selects the appropriate tool for the task and the input parameters are correctly formatted, it receives a score of 1 for that task; otherwise, the score is 0.
- **Result accuracy** (res acc): For search and recommendation instructions, we leverage the rank r of the target product within the returned product list from the tool as the metric, formulated as:

$$\text{Res Acc} = \begin{cases} 1 - \frac{r-1}{10}, & \text{if } r \leq 10, \\ 0, & \text{if } r > 10. \end{cases} \quad \text{with } r \in \mathbb{N}^+ \tag{1}$$

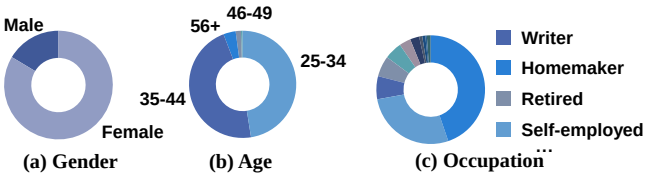
For review instructions, we assess the similarity between the agent's generated review and the user's actual review. We employ the sentence-transformer [37] model to compute the cosine similarity, yielding a res acc between 0 and 1.

• **Multi-turn track.** We believe it is crucial for Web agents to interact with users to receive feedback and continuously adjust their actions. Since using real humans for benchmark evaluation is impractical, we conduct **user simulators** based on LLMs to give real-time feedback. Specifically, we provide the LLM with user profiles, target product information, or ground-truth reviews to facilitate high-quality interactions between user simulators and Web agents. Please refer to Figure 15 for the details of the user simulator prompt.

In addition to the two metrics used in the single-turn track, we introduce an additional evaluation metric: **average steps**. This metric measures efficiency by counting the total number of actions taken to complete the task, encouraging the agent to accomplish users' tasks with minimal attempts.

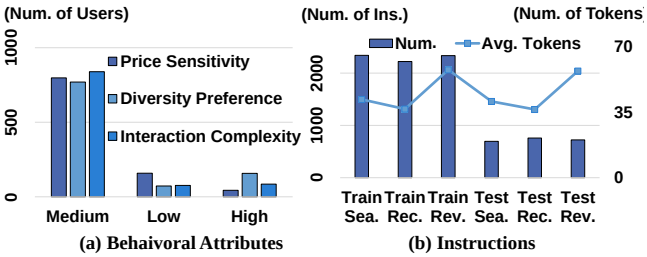
Table 2: Statistics of the PersonalWAB Benchmark.

	items	Train	Test
User	# Users	939	1,000
	# Avg. profile tokens	247	
	# Avg. behavior length	32	38
	# Avg. behavior tokens	7,597	9,270
Instruction	# Instructions	6,896	2,174
	# Avg. tokens	46	45
Product	# Products	8,236	
	# Avg. tokens	665	



(a) Gender (b) Age (c) Occupation

Figure 3: Distribution of users by gender, age, and occupation.



(a) Behaivoral Attributes (b) Instructions

Figure 4: (a) Distribution of behaviors by Price Sensitivity, Diversity Preference, and Interaction Complexity; (b) Statistics of the instructions on different tasks.

4 Benchmark Analysis

4.1 Statistic Analysis

We present the basic statistical information of our conducted PersonalWAB in Table 2. Since user profiles are generated in our benchmark, we analyze the diversity of all users and the consistency of each user's profile, to verify the reliability of PersonalWAB.

• **User statistics.** In Figure 3, we present the basic attributes of user profiles to illustrate the distribution. The data shows a reasonable spread across gender and age groups, while occupation categories cover a wide range of fields, ensuring diverse professional backgrounds in the dataset. The statistics in Figure 4 (a) highlight additional diversity in behavioral attributes such as *Price Sensitivity*, *Diversity Preference*, and *Interaction Complexity*. Most users fall into the "medium" category across these behavioral aspects, and the "high" and "low" categories are less frequent, which allows for testing both typical and edge cases in personalized tasks.

• **Instruction statistics.** We examined the number and average token length of different instructions in Figure 4 (b). It is observed that the recommendation instructions have the smallest number of tokens because the recommendation is an exploratory task and doesn't contain many user information requirements. The review instructions show slightly higher complexity than search and recommendation instructions, as they include many words for users to express their initial evaluations.

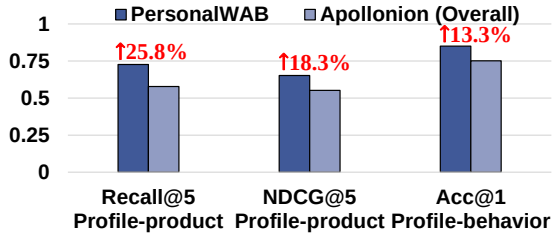


Figure 5: Results of profile consistency evaluation experiments. Our generated profiles align better with users' actual Web behaviors and interested products than Apollonion [5].

4.2 Profile Consistency Evaluation

Following [5], we conducted experiments on profile-behavior consistency and profile-product consistency to verify how well the generated profiles align with users' actual past Web behaviors and potentially interesting products. 1) **Profile-behavior consistency evaluation.** The task was to match a user profile with the user's past Web behaviors among those of other users. 2) **Profile-product consistency evaluation.** The task involved ranking a mixture of previously interacted (positive) and randomly sampled (negative) items for a user, based on their profile. The results in Figure 5 show that our constructed profiles provide substantial improvements in both profile-product and profile-behavior consistency evaluations compared to Apollonion [5], showcasing the enhanced distinctiveness and alignment of the profiles with the actual user behaviors. More details are in § A.1.

5 PUMA Framework

To enable the Web agent to effectively complete tasks following user instructions, we propose a novel PUMA framework, which consists of two key steps: Web function identification and function parameter generation (see Figure 6). First, we fine-tune an LLM (*e.g.,* LLaMa-2-7b [47]) with "instruction-function" pairs in the training set to identify the correct Web functions given a user instruction. Then, we generate the appropriate parameters for the identified function. To achieve this, PUMA first adopts a memory bank to store the users' long-term Web behaviors and utilizes a task-specific retrieval strategy to obtain the most relevant ones from the memory bank. With the obtained user behaviors and features, we instruct the LLM to generate the corresponding parameters. However, generating the appropriate parameters for the identified function poses a significant challenge due to the vast parameter space. To address this challenge, PUMA applies heuristic methods to construct pseudo-labels to further fine-tune the LLM and optimize parameter generation using DPO [36], ensuring superior alignment with user preferences.

5.1 Task-specific Memory Retrieval

• **Long-term memory bank.** The long-term memory bank is a storage system where we maintain a detailed record of each user's historical Web behaviors. For a user u, we store detailed information about their purchased products $h_{purchase}$ and the associated reviews h_{review}, collectively denoted as m. Specifically, the product details include attributes such as "title", "price", "store", and other relevant metadata, while the review details encompass the "rating",

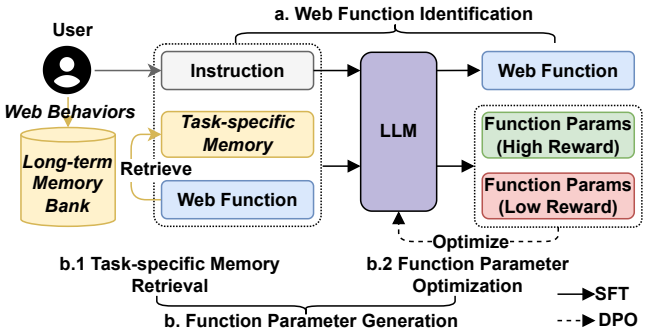


Figure 6: Illustration of the PUMA framework, consisting of two main steps: Web Function Identification and Parameter Generation, which includes Task-specific Memory Retrieval and Function Parameter Optimization.

"review title", and "comment" provided by the user. Formally, if the user u has purchased n products, the long-term memory M is represented as: $M = \{m_i \mid i = 1, 2, ..., n\}$.

• **Task-specific memory retrieval strategy.** The task-specific memory retrieval strategy is designed to extract relevant information from the long-term memory bank based on the user's current instruction and the identified function. When the user u provides an instruction i and the Web function f is determined, we first retrieve the top K memory entries by computing the cosine similarity between the instruction i and each memory m_j in the bank M. Then, based on the specific function f, we extract more targeted features from the retrieved memory. 1) If the Web function is related to search, we extract product details including the "product title", "category", "price", and "store". 2) If the function pertains to the recommendation, we retain the product "title", "category", and "parent ASIN" (product ID). 3) For review functions, only the user's past ratings and comments are kept. This process can be formally defined as:

$$M_i = \text{Extract}\left(\text{TopK}\left(M, \text{sim}(i, m_j), K\right), f\right). \tag{2}$$

M_i represents the task-specific memory constructed for instruction i. $\text{Extract}(\cdot, f)$ represents extracting targeted features based on the identified Web function f. The similarity $\text{sim}(i, m_j)$ is the cosine similarity between the instruction i and memory entry m_j.

5.2 Function Parameter Optimization

Given the task-specific memory M_i, the next step involves utilizing this memory to generate the Web function parameters. We begin by using SFT to equip the model with a foundational ability to generate reasonable parameters.

• **Heuristic fine-tuning for parameter generation.** The inputs for SFT are structured as a combination of the user instruction i, the task-specific memory M_i, and the identified Web function f. The labels are the Web function parameters, constructed using heuristic methods tailored to each Web function. 1) For the search function, we leverage ChatGPT [33] to generate textual queries based on the instruction, and memory. 2) For recommendations, the output consists of the most recent product ASINs from the same category found in the memory M_i. 3) For review functions, we use the actual review text provided by the dataset as the labels. These heuristics help construct meaningful pseudo-labels for parameter generation,

ensuring that the model learns to generate function parameters that are plausible and contextually appropriate for each Web function.

• **Diverse parameter sampling for pair-wise optimization.** After SFT equips the model with fundamental ability, we further enhance the model's performance through DPO [36] over diverse parameter candidates. We first generate a diverse set of function parameters with high-temperature sampling and beam search. These candidate parameters are then evaluated based on their result accuracy for instruction completion. For instruction i, we collect best-performing (p_i^{b}) and worst-performing (p_i^{w}) parameter pairs to construct the pair-wise preference data, which is formally defined as $\mathcal{D}_{\mathrm{DPO}}$:

$$\mathcal{D}_{\mathrm{DPO}} = \left\{ \left(p_i^{\mathrm{b}}, p_i^{\mathrm{w}}, x_i \right) \right\}, \tag{3}$$

where x_i represents the input, which includes the user instruction i, task-specific memory M_i, and Web function f.

We then apply DPO to optimize the fine-tuned model π_{ref} by encouraging it to generate function parameters similar to p_i^{b} and discouraging it from generating function parameters similar to p_i^{w}. The DPO loss is given by:

$$\mathcal{L}_{\mathrm{DPO}} = -\mathbb{E}\left[\log \sigma \left(\beta \log \frac{\pi_\theta(p^{\mathrm{b}} \mid x)}{\pi_{\mathrm{ref}}(p^{\mathrm{b}} \mid x)} - \beta \log \frac{\pi_\theta(p^{\mathrm{w}} \mid x)}{\pi_{\mathrm{ref}}(p^{\mathrm{w}} \mid x)} \right) \right], \tag{4}$$

where $\sigma(\cdot)$ is the sigmoid function, and β is a temperature-like parameter that controls the sensitivity of the model's preference to the log-ratio difference between the policy model π_θ for optimization and reference model π_{ref} derived from the SFT stage.

6 Experiments

We evaluate a range of baselines that employ different strategies for selecting and utilizing user history. These baselines are categorized into three groups: Memory Retrieval based Methods (No Memory, Random Memory, Last Memory, Relevant Memory), Enhanced Reasoning Methods (ReAct [56], Reflexion [42]), and Recommendation-Specific Memory Frameworks (Recmind [49], InteRecAgent [17]). The implementation details of baselines and our method are illustrated in § A.2 and § A.3. All methods use the same prompt template, with differences only in the memory component. The detailed prompt design is provided in Figure 16 and Figure 17.

6.1 Main Results

We evaluated baselines and our framework on both single-turn and multi-turn evaluation tracks.

6.1.1 Single-turn Track. The results on the single-turn track are shown in Table 3, highlighting several key insights: 1) It was observed that while search instructions had high function accuracy, the performance for recommendation instructions was poor. Further analysis revealed that many recommendation instructions were incorrectly assigned to the search function, as visualized in Figure 8 (b), indicating the great difficulty in function selection. 2) Methods incorporating relevant memory and ReAct show improved function accuracy, suggesting that retrieving relevant information and incorporating reasoning improve function selection. 3) The result accuracy for all baselines remains similar to the naive "No Memory" baseline, implying these methods fail to significantly enhance personalized task execution. 4) In contrast, PUMA achieves the highest function accuracy across tasks, with task-specific memory enabling the agent to focus on relevant behaviors and features, leading to higher result accuracy. Additionally, PUMA delivers the best overall performance while using shorter memory and a smaller LLM, highlighting the efficiency and effectiveness of our approach.

6.1.2 Multi-turn Track. The multi-turn track results (Table 4) offer valuable insights into how different methods handle complex interactions. 1) First, baselines perform better in search and recommendation tasks compared to the single-turn track, benefiting from multiple attempts and user feedback, while review tasks show minimal improvement, as the agents typically follow a straightforward flow with limited feedback opportunities. 2) The memory retrieval baselines follow similar trends to the single-turn track, with relevant memory improving function accuracy and result accuracy, but at the cost of additional steps. 3) ReAct and Reflexion perform worse than memory retrieval methods, requiring more steps and yielding lower accuracy. The complexity of these methods, which include reasoning and self-reflexion, seems to hinder task efficiency and accuracy with extra input token length. 4) RecMind also requires a higher number of steps, as it performs additional function calls, but struggles with instruction identification. InteRecAgent uses fewer steps due to its streamlined memory, but this simplification results in lower result accuracy. 5) Our Task-specific Memory method performs strongly, particularly in search and recommendation tasks. By extracting relevant information and filtering out redundant data, it enables more informed decisions with fewer steps. Although we did not evaluate the full PUMA approach due to model limitations in multi-turn settings, the results highlight the importance of task-specific memory in enhancing both efficiency and accuracy.

6.2 In-depth Analysis

We performed a comprehensive analysis of PUMA, including experiments on ablation study, memory length, efficiency, action transitions, multi-turn performance variation, function usage and outcome accuracy, search function implementation, and zero-shot and few-shot performance. Due to space limitations, we present the results for efficiency in this section, while the remaining analyses are provided in § A.4.

6.2.1 Analysis on efficiency. In real-world applications, task completion time is crucial for delivering a smooth user experience. To evaluate it, we measured the time taken to complete the user instruction in the single-turn track. Each method was tested on 100 randomly selected tasks, and the average completion time was calculated. The results in Figure 7 show that GPT-based methods have similar completion times, ranging from 6.5 to 6.9 seconds, due to memory processing overhead. In contrast, our PUMA framework significantly outperforms all baselines, with an average time of 2.8 seconds. This efficiency gain stems from PUMA's smaller model and compact memory structure, minimizing inference time, making it highly effective for real-world Web applications where quick response times are essential.

Table 3: Performance comparison between our proposed method, PUMA, and baselines in single-turn track. Bold numbers indicate the best performance in each column, while underlined numbers indicate the second-best performance.

Method (backbone)	Search		Recommendation		Review		Overall	
	Function Acc.	Res Acc	Function Acc.	Res Acc	Function Acc.	Res Acc	Function Acc.	Res Acc
No Memory (gpt-4o)	**1.000**	0.647	0.092	0.000	**1.000**	0.444	0.684	0.355
Random Memory (gpt-4o)	0.974	0.640	0.296	0.018	0.996	0.442	0.745	0.357
Last Memory (gpt-4o)	0.937	0.626	0.432	0.028	**1.000**	0.442	0.782	0.357
Relevant Memory (gpt-4o)	0.928	0.622	0.492	0.030	**1.000**	0.443	0.800	0.356
ReAct [56] (gpt-4o)	0.903	0.605	0.560	0.027	0.996	0.444	0.815	0.350
RecMind [49] (gpt-4o)	0.981	0.645	0.226	0.017	0.990	0.442	0.721	0.359
PUMA(gpt-4o)	**1.000**	0.649	0.939	0.048	**1.000**	0.449	0.979	0.373
PUMA(LLaMA-7B)	0.996	**0.652**	**0.987**	**0.054**	**1.000**	**0.538**	**0.994**	**0.406**

Table 4: Performance comparison between our proposed method, PUMA, and baselines in multi-turn track. F. Acc. represents function accuracy, R. Acc. stands for result accuracy, and Avg. Steps indicate the average number of steps taken by the agent to complete each instruction.

Method (backbone)	Search			Recommendation			Review			Overall		
	F. Acc.	R. Acc.	Avg. Steps	F. Acc.	R. Acc.	Avg. Steps	F. Acc.	R. Acc.	Avg. Steps	F. Acc.	R. Acc.	Avg. Steps
No Memory (gpt-4o)	0.996	0.656	2.398	0.096	0.000	2.420	**1.000**	0.446	2.019	0.685	0.358	2.280
Random Memory (gpt-4o)	0.999	0.680	4.193	0.703	0.042	4.474	**1.000**	0.448	2.007	0.896	0.380	3.564
Last Memory (gpt-4o)	0.996	0.676	4.229	0.708	0.045	4.252	**1.000**	0.449	2.007	0.897	0.381	3.498
Relevant Memory (gpt-4o)	0.996	0.686	4.233	0.715	0.042	4.564	0.999	0.448	2.008	0.899	0.383	3.609
ReAct [56] (gpt-4o)	0.996	0.674	4.657	0.218	0.013	5.468	0.974	0.448	2.129	0.718	0.369	4.098
Reflexion [42] (gpt-4o)	**1.000**	0.686	5.406	0.281	0.014	6.145	0.976	0.449	2.145	0.741	0.373	4.579
RecMind [49] (gpt-4o)	0.997	0.642	6.728	0.347	0.026	6.003	0.997	0.451	2.107	0.771	0.364	4.938
InteRecAgent [17] (gpt-4o)	0.999	0.642	3.110	0.618	0.022	3.008	**1.000**	0.447	2.001	0.867	0.362	2.706
PUMA (gpt-4o)	0.999	**0.720**	5.082	**0.984**	**0.052**	3.791	**1.000**	**0.453**	2.002	**0.994**	**0.399**	3.608

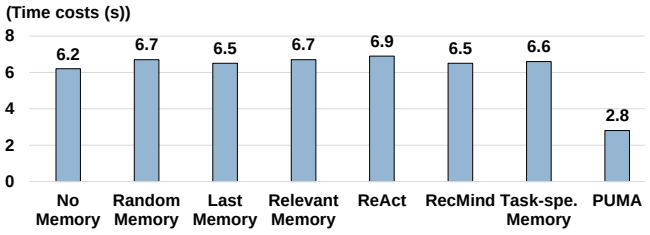


Figure 7: Comparison between the average task completion time (in seconds) for different methods.

7 Conclusion and Future Work

In this paper, we advanced general LLM-based Web agents into the era of personalized Web agents, aiming to offer users tailored and customized services. We formulated the task of LLM-empowered personalized Web agents and identified the goal of leveraging personalized user data to achieve personalized instruction understanding and action execution (Web function call). For training and evaluation, we constructed the first PersonalWAB benchmark on three personalized Web tasks. We proposed PUMA, a novel personalized alignment framework with task-specific memory and function parameter optimization strategies, to adapt LLMs to personalized Web agents. Extensive experiments on PersonalWAB demonstrate that PUMA consistently surpasses existing Web agents, aligning better with personalized user instructions and preferences. We believe that the task, benchmark, and framework for LLM-empowered personalized Web agents will broaden the research scope, introduce new challenges, and inspire novel methods in Web agent scenarios.

While our research lays the groundwork for personalized Web agents, several avenues for future exploration remain. First, we plan to extend PersonalWAB by incorporating more diverse task scenarios to further challenge and evaluate Web agents' personalization capabilities. Second, integrating more sophisticated user modeling techniques, such as dynamic preference learning, could enhance agents' adaptability to evolving user needs. Third, exploring user-in-the-loop settings presents an exciting opportunity to improve task execution by actively involving users in the process. This includes developing agents that can better integrate user feedback, proactively identify missing information, and engage with users to request necessary details, thereby enhancing the overall effectiveness and efficiency of task completion.

Although this work makes significant strides in personalized Web agents, it is important to note some limitations. Ethical and privacy considerations, along with the scope of our study, warrant further discussion. These aspects are elaborated in § B, where we provide a detailed exploration of challenges associated with user data usage, potential biases in personalization, and the current focus on shopping scenarios. While our approach demonstrates strong adaptability, extending it to broader Web environments presents additional complexities that require further investigation.

Acknowledgments

The work described in this paper was supported by the Research Grants Council of Hong Kong (PolyU/15209724, PolyU/15207821, PolyU/15207122, PolyU/15213323) and PolyU internal grants (BDWP).

References

[1] Mahyar Abbasian, Iman Azimi, Amir M. Rahmani, and Ramesh Jain. 2024. Conversational Health Agents: A Personalized LLM-Powered Agent Framework. arXiv:2310.02374

[2] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2019. The Unfairness of Popularity Bias in Recommendation. arXiv:1907.13286 [cs.IR] https://arxiv.org/abs/1907.13286

[3] Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. 2018. Equity of Attention: Amortizing Individual Fairness in Rankings. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. doi:10.1145/3209978.3210063

[4] Sanxing Chen, Sam Wiseman, and Bhuwan Dhingra. 2024. ChatShop: Interactive Information Seeking with Language Agents. arXiv:2404.09911

[5] Shangyu Chen, Zibo Zhao, Yuanyuan Zhao, and Xiang Li. 2024. Apollonion: Profile-centric Dialog Agent. arXiv:2404.08692

[6] 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. arXiv:2306.06070

[7] 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 the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 8795–8812.

[8] Hoa Dinh and Thien Khai Tran. 2023. EduChat: An AI-Based Chatbot for University-Related Information Using a Hybrid Approach. *Applied Sciences* (2023).

[9] Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How Capable Are Web Agents at Solving Common Knowledge Work Tasks? arXiv:2403.07718

[10] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783

[11] Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. 2024. Multimodal Web Navigation with Instruction-Finetuned Foundation Models. In *The Twelfth International Conference on Learning Representations*.

[12] Chongming Gao, Shiqi Wang, Shijun Li, Jiawei Chen, Xiangnan He, Wenqiang Lei, Biao Li, Yuan Zhang, and Peng Jiang. 2023. CIRS: Bursting Filter Bubbles by Counterfactual Interactive Recommender System. *ACM Transactions on Information Systems (TOIS)* 42, 1, Article 14 (aug 2023), 27 pages. doi:10.1145/3594871

[13] Izzeddin Gur, Hiroki Furuta, Austin V Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2024. A Real-World WebAgent with Planning, Long Context Understanding, and Program Synthesis. In *The Twelfth International Conference on Learning Representations*.

[14] 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. *arXiv preprint arXiv:2401.13919* (2024).

[15] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. Bridging Language and Items for Retrieval and Recommendation. *arXiv preprint arXiv:2403.03952* (2024).

[16] 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 *International Conference on Learning Representations*.

[17] Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2023. Recommender AI Agent: Integrating Large Language Models for Interactive Recommendations. arXiv:2308.16505

[18] Mingyu Jin, Qinkai Yu, Dong Shu, Chong Zhang, Lizhou Fan, Wenyue Hua, Suiyuan Zhu, Yanda Meng, Zhenting Wang, Mengnan Du, and Yongfeng Zhang. 2024. Health-LLM: Personalized Retrieval-Augmented Disease Prediction System. arXiv:2402.00746

[19] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. 197–206.

[20] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2024. Language models can solve computer tasks. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Article 1723, 30 pages.

[21] Milton King and Paul Cook. 2020. Evaluating Approaches to Personalizing Language Models. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*. European Language Resources Association, 2461–2469.

[22] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. *arXiv preprint arXiv:2401.13649* (2024).

[23] Ishita Kumar, Snigdha Viswanathan, Sushrita Yerra, Alireza Salemi, Ryan A. Rossi, Franck Dernoncourt, et al. 2024. LongLaMP: A Benchmark for Personalized Long-form Text Generation. arXiv:2407.11016

[24] Gibbeum Lee, Volker Hartmann, Jongho Park, Dimitris Papailiopoulos, and Kangwook Lee. 2023. Prompted LLMs as Chatbot Modules for Long Open-domain Conversation. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, 4536–4554.

[25] Zhenyang Li, Fan Liu, Yinwei Wei, Zhiyong Cheng, Liqiang Nie, and Mohan Kankanhalli. 2024. Attribute-driven Disentangled Representation Learning for Multimodal Recommendation. In *Proceedings of the 32nd ACM International Conference on Multimedia*. ACM, 9660–9669.

[26] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 2356–2362.

[27] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, and Percy Liang. 2018. Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration. In *International Conference on Learning Representations*.

[28] Junru Lu, Siyu An, Mingbao Lin, Gabriele Pergola, Yulan He, Di Yin, Xing Sun, and Yunsheng Wu. 2023. MemoChat: Tuning LLMs to Use Memos for Consistent Long-Range Open-Domain Conversation. arXiv:2308.08239

[29] Xing Han Lù, Zdeněk Kasner, and Siva Reddy. 2024. WebLINX: Real-World Website Navigation with Multi-Turn Dialogue. arXiv:2402.05930

[30] Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. 2018. Training Millions of Personalized Dialogue Agents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2775–2779.

[31] Fatemehsadat Mireshghallah, Vaishnavi Shrivastava, Milad Shokouhi, Taylor Berg-Kirkpatrick, Robert Sim, and Dimitrios Dimitriadis. 2022. UserIdentifier: Implicit User Representations for Simple and Effective Personalized Sentiment Analysis. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 3449–3456.

[32] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021).

[33] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. 2024. GPT-4 Technical Report. arXiv:2303.08774.

[34] Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents. arXiv:2408.07199

[35] Alec Radford. 2018. Improving language understanding by generative pre-training. (2018).

[36] 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 *Thirty-seventh Conference on Neural Information Processing Systems*.

[37] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[38] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3982–3992. doi:10.18653/v1/D19-1410

[39] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2024. LaMP: When Large Language Models Meet Personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 7370–7392.

[40] Shady Shehata, David Santandreu Calonge, Philip Purnell, and Mark Thompson. 2023. Enhancing Video-based Learning Using Knowledge Tracing: Personalizing Students' Learning Experience with ORBITS. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*. Association for Computational Linguistics, 100–107.

[41] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of Bits: An Open-Domain Platform for Web-Based Agents. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 3135–3144.

[42] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Article 377.

[43] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 3–18.

[44] Nikita Soni, Matthew Matero, Niranjan Balasubramanian, and H. Andrew Schwartz. 2022. Human Language Modeling. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics, 622–636.
[45] Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. 2024. AdaPlanner: adaptive planning from feedback with language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Article 2537, 44 pages.
[46] Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. 2022. META-GUI: Towards Multi-modal Conversational Agents on Mobile GUI. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 6699–6712.
[47] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).
[48] Yu-Min Tseng, Yu-Chao Huang, Teng-Yun Hsiao, Wei-Lin Chen, Chao-Wei Huang, Yu Meng, and Yun-Nung Chen. 2024. Two Tales of Persona in LLMs: A Survey of Role-Playing and Personalization. arXiv:2406.01171
[49] Yancheng Wang, Ziyan 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*. Association for Computational Linguistics, 4351–4364.
[50] Charles Welch, Chenxi Gu, Jonathan K. Kummerfeld, Veronica Perez-Rosas, and Rada Mihalcea. 2022. Leveraging Similar Users for Personalized Language Modeling with Limited Data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 1742–1752.
[51] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. 2023. TidyBot: Personalized Robot Assistance with Large Language Models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 3546–3553.
[52] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, et al. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv:2308.08155
[53] Kevin Xu, Yeganeh Kordi, Tanay Nayak, Ado Asija, Yizhong Wang, Kate Sanders, Adam Byerly, Jingyu Zhang, Benjamin Van Durme, and Daniel Khashabi. 2024. Tur[k]ingBench: A Challenge Benchmark for Web Agents. arXiv:2403.11905
[54] Nancy Xu, Sam Masling, Michael Du, Giovanni Campagna, Larry Heck, James Landay, and Monica Lam. 2021. Grounding Open-Domain Instructions to Automate Web Support Tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 1022–1032.
[55] 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*, Vol. 35. Curran Associates, Inc.
[56] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*.
[57] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing Dialogue Agents: I have a dog, do you have pets too?. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2204–2213.
[58] Ziniu Zhang, Shulin Tian, Liangyu Chen, and Ziwei Liu. 2024. MMInA: Benchmarking Multihop Multimodal Internet Agents. arXiv:2404.09992
[59] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. GPT-4V(ision) is a Generalist Web Agent, if Grounded. In *Forty-first International Conference on Machine Learning*.
[60] Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2024. Synapse: Trajectory-as-Exemplar Prompting with Memory for Computer Control. In *The Twelfth International Conference on Learning Representations*.
[61] Hanxun Zhong, Zhicheng Dou, Yutao Zhu, Hongjin Qian, and Ji-Rong Wen. 2022. Less is More: Learning to Refine Dialogue History for Personalized Dialogue Generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 5808–5820.
[62] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. 2024. WebArena: A Realistic Web Environment for Building Autonomous Agents. *ICLR* (2024).

A Details

A.1 Details of Profile Evaluation Experiments

Profile-behavior consistency evaluation. Given a specific user profile, the task is to identify the correct user from a group of candidate users, consisting of the true user and several negative users. Each candidate user is represented by their behavior sequence, and the objective is to determine which candidate aligns best with the given profile. The evaluation metric used is top-1 accuracy, indicating the ability of the profile to distinctly and accurately match the correct user based on their behaviors.

Profile-product consistency evaluation. In this task, a given user profile is used to rank a set of candidate items, which include a mixture of positive items (previously interacted with by the user) and negative items (randomly sampled from an item pool). The objective is to prioritize positive items over negative items, leveraging the user profile for accurate ranking. The task is evaluated with NDCG@5 and Recall@5, which measure the profile's ability to reflect the user's preferences.

We adopt the same setting with [5], set the number of positive samples to 1 and 3, and negative samples to 4 and 7 in the user prediction and recommendation tasks, respectively. To ensure a fair evaluation, both experiments are conducted in separate, isolated sessions. Experiments are conducted with `gpt-4o-mini-2024-07-18`, and results in Figure 5 show our profile exhibits significant improvements across both tasks, indicating that our profiles are more distinct and better aligned with user behaviors.

A.2 Details of Baselines

We evaluate a range of baselines that employ different strategies for selecting and utilizing user Web behavioral history. These baselines are categorized into three groups: Memory Retrieval Methods, Enhanced Reasoning Methods, and Recommendation-Specific Memory Frameworks.

Memory Retrieval Methods. We include various simple memory mechanisms as baselines, aiming to explore different strategies for selecting and utilizing user history. This helps us understand how each memory selection technique impacts task performance. 1) No Memory: The agent performs tasks without accessing any user history, relying solely on the current instruction. 2) Random Memory: This approach randomly selects behaviors from the user's history. 3) Last Memory: Uses only the most recent behaviors from the user's history, focusing on the assumption that recent context is more relevant for current instruction. 4) Relevant Memory: Selects past behaviors based on cosine similarity with the current instruction, aiming to filter out the most contextually relevant details for the task.

Enhanced Reasoning Methods. We also tested frameworks designed to enhance the agent's reasoning and decision-making capabilities. 1) ReAct: Proposed by [56], this framework instructs the language model to think before taking an action and generate "Thought: some reasoning Action: some JSON format action argument" to interact, enabling the model to deliberate over the information available and decide on the most appropriate action. 2) Reflexion: Reflexion [42] builds on frameworks of ReAct by adding a self-evaluation phase, where the agent reviews and analyzes its previous actions and outcomes before proceeding. This process allows the agent to recognize mistakes, reassess decisions, and refine its strategy in subsequent interactions. We tested this baseline only in the multi-turn track, where the agent treats each user message as feedback for Reflexion and adjustment.

Table 5: Alation study on key components of PUMA in single-turn track.

Method	Search		Recommendation		Review		Overall	
	Function Acc	Result Acc	Function Acc	Result Acc	Function Acc	Result Acc	Function Acc	Result Acc
PUMA	0.996	0.652	0.987	0.054	1.000	0.538	0.994	0.406
w/o Task-specific Memory	0.990	0.643	0.992	0.008	1.000	0.496	0.994	0.373
w/o SFT	1.000	0.000	0.983	0.000	1.000	0.160	0.994	0.054
w/o DPO	0.996	0.648	0.987	0.047	1.000	0.529	0.994	0.399

Table 6: Performance comparison of different memory token lengths in PUMA.

Memory Length	Search		Recommendation		Review		Overall	
	Function Acc	Result Acc	Function Acc	Result Acc	Function Acc	Result Acc	Function Acc	Result Acc
256	0.997	0.651	0.985	0.019	1.000	0.530	0.994	0.391
512	0.991	0.648	0.988	0.032	1.000	0.531	0.993	0.395
768	0.996	**0.652**	0.987	**0.054**	1.000	**0.538**	0.994	**0.406**

Recommendation-Specific Memory Frameworks. Recommendation tasks are inherently personalized, as they rely on a deep understanding of user preferences and behaviors [12, 17, 19, 25, 49]. Given this, we include baselines that leverage memory mechanisms developed for recommendation agents, assessing their ability to enhance personalization in our context. 1) RecMind: An LLM-powered agent designed for general recommendation purposes [49], consists of two parts of memory, personalized memory, and world knowledge. Personalized Memory includes individualized user information, such as their reviews or ratings for a particular item. World Knowledge consists of item metadata information and real-time information accessed with a Web search function. In our setup, we retain the personalized memory containing user reviews and ratings, and we incorporate an additional function to enable RecMind to access detailed product information. 2) InteRecAgent: Proposed by [17], this framework uses LLMs as the core reasoning engine while utilizing recommender models as functions for interactive recommendations. Its memory structure includes a candidate bus (which stores current item candidates) and a user profile that captures three facets of user preferences: "like", "dislike", and "expect". We adopt the user profile memory in our experiments and allow the agent to update this profile at the end of each task. As the user profile is synthesized by LLMs based on conversation history with the user, we evaluate this method only in the multi-turn setting, where ongoing dialogue allows for continuous adaptation of the user profile.

A.3 Implementation Details

Benchmark. We utilize `gpt-4o-mini-2024-07-18` for generating user profiles, as it excels at extracting detailed user preferences, particularly in capturing brand preferences. For user instruction creation, we employ `claude-3-5-sonnet@20240620`, selected for its ability to produce instructions in a natural and human-like tone. In multi-turn track, `gpt-4o-mini-2024-07-18` is also used to simulate user messages, as it follows the instructions better to give user messages.

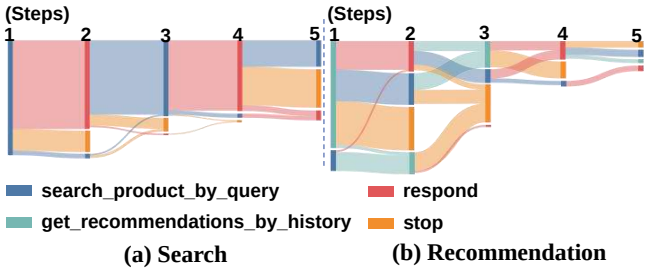
Baselines. We use `gpt-4o-mini-2024-07-18` as the base language model across all baseline methods. For memory retrieval baselines, we set the memory length to 50 behaviors for the single-turn track and 20 behaviors for the multi-turn track, allowing additional input length for user messages and function results. For the Relevant Memory method, we calculate cosine similarity using the sentence-transformer [37] to identify relevant behaviors. The ReAct baseline is combined with the Last Memory approach to ensure that reasoning processes have recent context, and we further extend this with a Reflexion mechanism for multi-turn scenarios. For RecMind, the memory length is set to 400 behaviors, as it only contains user reviews and ratings, and we added an extra "get_product_details_by_asin" function for the agent to retrieve detailed product information. In the InteRecAgent setup, we first construct the memory using historical behaviors and the training dataset before evaluating performance on the test set.

PUMA. We generate the function parameters for search using `gpt-4o-mini-2024-07-18` as the initial SFT labels. In the function parameter optimization phase, we fine-tune the LLaMA2-7B [47] model with LoRA [16] using 4 × 24GB NVIDIA A5000 GPUs. The learning rate is set to 4e-3 for the SFT and 5e-5 during the DPO stage, with a batch size of 1 per GPU. Due to the maximum sequence length we can afford during training is limited, we constrain the memory token length to 256, 512, and 768 tokens. To generate diverse function parameters, we set a temperature of 1.5 to increase output variability and use a beam search with a beam size of 10.

A.4 More Analysis

A.4.1 Ablation study. We conducted an ablation study (see Table 5) to assess the impact of PUMA's key components. First, removing the memory leads to a significant drop in result accuracy across all tasks, highlighting the importance of memory in retaining relevant information for function parameter generation. Second, when memory is retained but the SFT phase is removed, result accuracy dramatically declines. This indicates that without fine-tuning, the model struggles to generate function parameters that align with user needs. Finally, removing the DPO phase results in a slight performance decrease, suggesting that DPO plays a crucial role in aligning the model with user preferences, and improving the quality of function parameters.

A.4.2 Analysis on memory length. We evaluated the impact of different memory token lengths on our framework's performance across tasks. The experiment measured both function accuracy and result accuracy with varying memory sizes. The results in Table 6 indicate that increasing memory length has minimal impact



(a) Search (b) Recommendation

Figure 8: Transitions of the agent's actions in multi-turn search and recommendation tasks. Each color represents a specific function. The horizontal axis shows interaction steps, while the width of each color band indicates the proportion of the agent's focus on that action. The flow between steps illustrates how the agent adapts its strategy over steps.

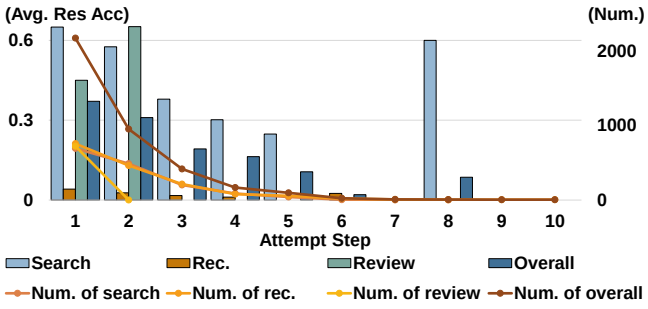


Figure 9: Analysis of the agent's performance across multiple attempts in multi-turn track.

on function accuracy, with the model maintaining similar performance regardless of memory size. However, memory length has a significant effect on result accuracy, particularly in recommendation tasks. Shorter memory lengths reduce the number of stored products, limiting the model's ability to select appropriate product IDs, which leads to a noticeable drop in recommendation accuracy. In contrast, search and review tasks are less sensitive to memory length changes, as the agent relies more on information from the instruction rather than the memory. This reduced dependence on memory in these tasks may also limit the model's potential to improve performance further.

A.4.3 Analysis on action transitions. We collected PUMA's actions in each interaction turn within the multi-turn track. Review instructions were removed, as the agent typically completes them in just two steps. The results, visualized in Figure 8, provided the following insights. 1) For search instructions, the agent tends to alternately call "search" and "respond" functions. It is reasonable as the agent could receive user feedback via the "respond" function and thus adjust its search action. 2) The bond for the recommendation instructions is more entangled in Figure 8 (b), indicating a more complex action transition. This underlines the challenge in multi-turn recommendation tasks, where correctly identifying user intent and dynamically adjusting actions are more difficult than in straightforward tasks like search.

A.4.4 Analysis of multi-turn performance variation. We evaluated the agent's performance over multiple attempts in the multi-turn track. For each instruction, we measured the "Res Acc," and the number of solved tasks as the number of attempt steps increased. The results are shown in Figure 9, and we had the following observations. 1) The high task count within the first five attempts indicates that most tasks are completed early on. The "review" task is typically finished within the first two attempts, as there's little need for the agent to interact with users about the review requirements. 2) "Res Acc" is high during the initial attempts and declines with each subsequent attempt. This is because the easier tasks can be solved in a few attempts, leaving the more difficult tasks for later. 3) There are a few outliers where tasks achieve higher "Res Acc" in later steps. However, these cases are rare, involving only one or two tasks that result in the outlier. 4) The declined "Res Acc" also suggested that the agent struggles to leverage user feedback in later attempts effectively. This may be due to the lack of multi-turn training data, preventing us from tuning the agent accordingly.

Table 7: Single-turn performance comparison of different methods in terms of function accuracy (F. Acc.), result accuracy (R. Acc.), and outcome accuracy (O. Acc.) in search and recommendation.

Method	Search			Recommendation		
	F. Acc.	R. Acc.	O. Acc.	F. Acc.	R. Acc.	O. Acc.
No Memory	1.000	0.647	0.647	0.092	0.000	0.155
Random Memory	0.974	0.640	0.642	0.296	0.018	0.159
Last Memory	0.937	0.626	0.632	0.432	0.028	0.161
Relevant Memory	0.928	0.622	0.631	0.492	0.030	0.159
ReAct [56]	0.903	0.605	0.628	0.560	0.027	0.160
RecMind [49]	0.981	0.645	0.647	0.226	0.017	0.152
PUMA	**1.000**	**0.649**	**0.649**	**0.939**	**0.048**	**0.164**

Table 8: Comparison of search result accuracy using BM25 and Dense retrieval methods in single-turn track.

Method	Search (Result Accuracy)	
	BM25	Dense Retrieval
No Memory	0.647	0.502
Random Memory	0.640	0.504
Last Memory	0.626	0.498
Relevant Memory	0.622	0.499
ReAct [56]	0.605	0.496
RecMind [49]	0.645	0.498
PUMA	**0.649**	**0.506**

A.4.5 Analysis on function usage and outcome accuracy. In real-world applications, users care about whether the system retrieves relevant results rather than the specific function it employs. While search and recommendation serve different roles, their distinction may not always be meaningful from a user's perspective. For real users, the ultimate priority lies in the relevance of the returned items rather than the specific function used by the agent.

To better align with this user-centric goal, we introduce Outcome Accuracy (O. Acc.), a metric that evaluates the correctness of the returned results independent of the function used for search and recommendation tasks. As shown in Table 7, function accuracy (F. Acc.) and result accuracy (R. Acc.) vary significantly between search and recommendation tasks. However, Outcome Accuracy provides a more balanced perspective on the effectiveness of different methods. Our results show that PUMA achieves the highest

Table 9: Performance comparison in zero-shot and few-shot scenarios in single-turn track.

Method	Search		Recommendation		Review		Overall	
	Function Acc.	Result Acc.	Function Acc.	Result Acc.	Function Acc.	Result Acc.	Function Acc.	Result Acc.
No Memory	**1.000**	0.684	0.050	0.000	**1.000**	0.388	0.625	0.328
Random Memory	0.974	0.684	0.301	0.060	0.996	0.391	0.715	0.352
Last Memory	**1.000**	0.683	0.314	0.058	**1.000**	0.396	0.730	0.353
Relevant Memory	0.928	0.675	0.405	0.078	**1.000**	**0.397**	0.743	0.358
ReAct [56]	0.945	0.675	0.475	0.080	0.996	0.393	0.774	0.358
RecMind [49]	0.973	0.680	0.320	0.063	0.996	0.394	0.722	0.354
PUMA	**1.000**	**0.686**	**0.892**	**0.090**	**1.000**	0.396	**0.958**	**0.366**

Outcome Accuracy across both search and recommendation tasks, demonstrating its ability to deliver relevant results without being constrained by function selection. By incorporating this metric, we offer a more comprehensive evaluation that prioritizes relevance over strict function adherence, reflecting real-world user needs.

A.4.6 Analysis on search function implementation. In our benchmark, the implementations of both the search and recommendation functions are flexible and not restricted to a specific method. Our focus is not on optimizing these functions but on evaluating the agent's ability to effectively utilize the provided function in a personalized setting.

We conducted an alternative retrieval experiment in the single-turn track, replacing BM25 with a dense retrieval model based on Sentence-BERT [38]. The results in Table 8 indicated that while dense retrieval captures richer semantic representations, it also introduces noise by embedding extensive product details, leading to a slight degradation in result accuracy across all baselines. And different retrieval methods may impact performance, but the overall trends remain consistent. Despite these variations, our PUMA framework consistently outperformed all baselines, regardless of the retrieval model used. Given the modular nature of our benchmark, future work could explore alternative retrieval models and different recommendation strategies, further analyzing how function-level improvements lead to final performance.

A.4.7 Analysis on zero-shot and few-shot performance. Since our user sampling is fully random, some users naturally have limited historical data, leading to zero-shot and few-shot scenarios. To evaluate performance in such cases, we analyzed 139 users with fewer than 10 historical records (16.2% of the test set) in the single-turn track. Results in Table 9 show task-dependent effects: Search performance remains stable or improves slightly due to reduced irrelevant information, while Recommendation performance also improves, as limited memory simplifies retrieval. However, Review tasks decline, as the lack of past reviews hinders personalized responses. Despite these variations, PUMA consistently outperforms all baselines, achieving the highest accuracy across tasks, particularly excelling in Recommendation scenarios. These results demonstrate that PUMA remains effective even in zero-shot and few-shot conditions, highlighting its adaptability when user history is sparse.

B Discussion

B.1 Ethical and Privacy Considerations

The integration of personalized data in Web agents plays a crucial role in improving task efficiency and relevance. By leveraging user history, preferences, and past interactions, personalized Web agents can better align with individual needs and enhance task completion. However, the use of such data raises important ethical and privacy concerns that must be carefully considered.

One primary ethical concern is fairness in personalization. If personalization is not properly managed, it may introduce or amplify biases in decision-making processes. For example, recommendation algorithms might reinforce popularity bias [2], leading to a concentration of exposure on frequently suggested items while limiting diversity. Additionally, personalization can inadvertently discriminate against certain user groups if historical biases exist in the training data [3]. To mitigate these risks, fairness-aware personalization techniques and diversity-promoting strategies should be considered when designing personalized Web agents.

From a privacy perspective, handling user history introduces risks related to data security. Users may inadvertently disclose sensitive information, such as browsing behaviors, or purchase histories, which could be exploited if not properly secured. To ensure privacy protection, future work about personalized Web agents should incorporate privacy-preserving techniques, reducing the risk of data leakage while maintaining personalization benefits.

B.2 Scope of This Work

We chose the shopping domain as a representative and practically significant application to demonstrate the capabilities of personalized Web agents. Shopping platforms provide structured user behavior data and diverse interactions, such as search, recommendation, and product reviews, making them ideal testbeds for evaluating personalization strategies. Additionally, personalization in shopping directly impacts user experience and decision-making, highlighting the effectiveness of adaptive Web agents.

While our benchmark is designed on shopping-related Web functions, the framework introduced in this work is generalizable and can be extended to a broader range of Web applications, such as news recommendation, and social media content curation. These domains, like shopping, rely on user behavior modeling to enhance relevance and improve interaction quality.

C Prompting Details

The prompt template for profile generation is shown in Figure 10 and Figure 11. The prompt template for instruction generation is shown in Figure 12, Figure 13, and Figure 14. The prompt template for the user simulator is shown in Figure 15. Last, the prompt templates in task executing are shown in Figure 16 and Figure 17.

User Profile Generation

You will act as an online shopper.
Given your time-series historical purchase information and corresponding reviews, you need to summarize and choose the most accurate and relevant option best describing you.
During summarizing you should obey the following procedures:
First, summarize your basic information, and choose or fill the most accurate and relevant option for each category.
The categories and options are as follows:

- **Gender**: <GENDER>.
- **Age**: <AGE>.
- **Occupation**:<OCCUPATION>.
- **Price Sensitivity**: <PRICE SENSITIVITY>.
- **Shopping Interest**: Summarize the product information.
- **Brand Preference**: Choose from the product information, and only keep brand names.

Second, summarize your personal preferences across the following aspects:

- **Diversity Preference**:
  Do you prefer trying new things or sticking to familiar products? Choose from <DIVERSITY>.
- **Interaction Complexity**
  Do you prefer simple and concise interactions, or do you enjoy detailed and thorough exchanges? Choose from <INTERACTION>.
- **Tone and Style**
  Summarize your overall emotional tone, speaking style, and expressive characteristics when giving reviews. Keep keywords.
- **Item Reference**
  Summarize your tendency to refer to specific products or brands in your reviews, like purchase history, shopping cart, or recommendations from friends. Keep keywords.
- **Focus Aspect**
  What aspects of products do you pay more attention to? Choose from <FOCUS ASPECT> or summarize others from reviews. Keep keywords.

In the end, arrange all the above aspects using the JSON format, with each aspect as an individual key.
Do not include any additional information or explanations and stay grounded.
Your History:
<HISTORY>

Figure 10: User profile generation.

Details of Choices in Profile Generation

- **<Gender>:**
  [Female, Male].
- **<AGE>:**
  [Under 18, 18-24, 25-34 , 35-44 , 45-49 , 50-55 , 56+].
- **<OCCUPATION>:**
  [Academic/Educator, Artist, Clerical/admin, College/grad student, Customer service, Doctor/health care, Executive/managerial, Farmer, Homemaker, K-12 student, Lawyer, Programmer, Retired, Sales/Marketing, Scientist, Self-employed, Technician/Engineer, Tradesman/Craftsman, Unemployed, Writer, Other].
- **<PRICE SENSITIVITY>:**
  ["High": "A Price-Conscious Shopper who is very sensitive to cost and seeks the best deals.",
  "Medium": "A Balanced Buyer who considers price but also values quality and features.",
  "Low": "A Value-Driven Consumer who prioritizes quality and features over price."].
- **<DIVERSITY>:**
  ["High": "A Highly Adventurous Explorer eager to discover diverse products across categories. They often seek recommendations, and purchase a wide variety of items with varying ratings, and the user's own ratings may often differ from the average. Their reviews are detailed and enthusiastic, reflecting their unique tastes and enjoyment of variety.",
  "Medium": "A Balanced Seeker who enjoys trying new products but also values familiarity. They appreciate targeted recommendations, purchase a moderate number of items with solid ratings and a reasonable number of ratings, and their reviews balance detailed feedback with concise, practical comments.",
  "Low": "A Meticulously Selective Buyer who sticks to tried-and-true products, showing little interest in new options. They purchase fewer items, favoring those with high ratings and a large number of ratings. Their own ratings are often very close to or slightly above the average, and their reviews are thoughtful and focused on familiar products."].
- **<INTERACTION>:**
  ["High": "A Thorough Conversationalist who enjoys detailed discussions, exploring all aspects of a product or service. They provide extensive reviews and value comprehensive support, engaging in multiple rounds of communication.",
  "Medium": "A Moderate Engager who balances simplicity with detail. They prefer clear communication but can engage in detailed exchanges when necessary. They provide reviews that are a mix of concise observations and some detailed insights, especially if they have strong feelings about a product.",
  "Low": "A Minimalist Interactor who values simplicity and efficiency. They prefer quick, straightforward interactions and leave brief, to-the-point reviews, focusing only on essential product aspects."].
- **<FOCUS ASPECT>:**
  ["Average Rating", "Number of Ratings", "Price", "Store", "Material", "Size", "Weight", "Brand"].

Figure 11: Details of choices in user profile generation.

Search Instruction Generation

You will act as an online shopper.
Your Profile:
<PROFILE>
You are looking for a product similar to the following product:
<PRODUCT>
You want to find a similar product, but you are not looking for an exact match.
Generate a search request that is somewhat vague, reflecting your preferences and personalities without revealing the complete details of the target product.
Rules:

- <DIVERSITY>.
- <INTERACTION>.
- You pay more attention to <FOCUS_ASPECT> of products, make sure to include some of them in the search request.
- Ensure the search request aligns with your overall tone and style: <TONE_AND_STYLE>.
- Do not repeat the exact information in your profile or product. Instead, use your own words to convey the same information.
- Try to make the request as natural as possible and stick to the personalities in your profile.
- Do not include any additional information or explanations and stay grounded.
- Do not hallucinate information that is not provided,
- No more than <NUM> words.

Figure 12: Search instruction generation.

Recommendation Instruction Generation

You will act as an online shopper.
Your Profile:
<PROFILE>
You have recently shown interest in the following type of product:
<PRODUCT>
Now, you're exploring options that could match your overall tastes, but you're not sure exactly what you're looking for.
Generate a recommendation request that reflects your general preferences and style, but leaves room for flexibility and discovery.
Rules:

- <DIVERSITY>.
- <INTERACTION>.
- You value <FOCUS_ASPECT> of products, but keep the request open-ended to allow for a variety of recommendations.
- Ensure the recommendation request aligns with your overall tone and style: <TONE_AND_STYLE>.
- Do not restate your profile or the product. Use different words or hints to convey your preferences.
- Avoid being too specific or precise in your request.
- Try to make the request as natural as possible and stick to the personalities in your profile.
- Do not include any additional information or explanations and stay grounded.
- Do not hallucinate information that is not provided,
- No more than <NUM> words.

Figure 13: Recommendation instruction generation.

Review Instruction Generation

You will act as an online shopper.
Your Profile:
<PROFILE>,
You have recently purchased the following product:
<PRODUCT>,
Your feelings about the product are:
<REVIEW>,
Now, You want to write a review.
Generate a review request to ask for assistance to create a complete review that reflects your preferences and typical review style.
Rules:

- <INTERACTION>.
- You value <FOCUS_ASPECT> of products, but keep the request open-ended to allow for a variety of recommendations.
- Ensure the review request aligns with your overall tone and style: <TONE_AND_STYLE>.
- Do not simply restate your profile or feedback verbatim. Instead, paraphrase and expand to reflect a more comprehensive review.
- Try to make the request as natural as possible and stick to the personalities in your profile.
- Do not include any additional information or explanations and stay grounded.
- Do not hallucinate information that is not provided.
- No more than <NUM> words.

Figure 14: Review instruction generation.

User Simulation Instruction

You are a user interacting with a personalized shopping agent.
Your Profile:
<PROFILE>
You have purchased the following product:
<PRODUCT>
(In review tasks:) Your review is as follows:
<REVIEW>
The shopping agent will help you complete your shopping requests.
Rules:

- Just generate one line at a time to simulate the user's message.
- Do not hallucinate information that is not provided.
- Do not give additional instructions or ask questions, only respond to the agent's questions.
- Do not provide any specific product details.
- Do not repeat the exact information in your profile or product. Instead, use your own words to convey the same information.
- Try to make the conversation as natural as possible and stick to the personalities in your profile.
- If the result is not satisfactory, you can express your dissatisfaction and provide clues to help the agent understand your preferences.

Figure 15: User simulation instruction.

Prompt in Single-Turn Track

As a personalized shopping agent, you can help users search for products, recommend products, or complete their reviews.
Rules:

- The user will provide user_id and a request.
- You need to use the most appropriate tool to find the product or fill the review that matches the user's request.
- You are not allowed to interact with the user. Make the best tool call based on the user's request.
- You have only one chance to make a tool call, so make sure you have the best input for the tool.
- The tool will be provided, you need to use the tool and provide the most appropriate input for the tool. Do not use other tools.
- Formulate the best input for the tool based on the user's request and the memory provided.

Memory:
(Depends on the method.)
Functions to use:
(Available functions and their descriptions.)

Figure 16: Prompt template for all methods in single-turn track.

Prompt in Multi-Turn Track

As a personalized shopping agent, you can help users search for products, recommend products, or complete their reviews.
Rules:

- The user will provide user_id and a request.
- You need to use the tools to find the product or fill the review that matches the user's request.
- The tool will be provided, you need to use the tool and provide the most appropriate input for the tool. Do not use any other tools.
- Formulate the best input for the tool based on the user's request and the memory provided.
- You are allowed to interact with the user by 'respond' to ask for more information or feedback, but steps are limited, and less steps are preferred.
- Your main goal is to help the user complete the task as accurately and efficiently as possible, do not keep responding to the user, focus on making the better tool calls.
- The evaluation will be based on the ranking of the target product in search and recommendation tasks, and the similarity of the review in the review task.
- When you think you have found the best input for the task tool calls, you can end the task by making a 'stop' call.
- You should not make up any information or knowledge not provided from the user or the tools, or give subjective comments or recommendations.
- You should at most make one tool call at a time, and if you take a tool call, you should not respond to the user at the same time. If you respond to the user, you should not make a tool call.

Memory:
(Depends on the method.)
Functions to use:
(Available functions and their descriptions.)

Figure 17: Prompt template for all methods in multi-turn track.