Advancing and Benchmarking Personalized Tool Invocation for LLMs

Anonymous ACL submission

Abstract

Tool invocation is a crucial mechanism for extending the capabilities of Large Language Models (LLMs) and has recently garnered significant attention. It enables LLMs to solve complex problems through tool calls while accessing up-to-date world knowledge. However, existing work primarily focuses on the fundamental ability of LLMs to invoke tools for problem-solving, without considering personalized constraints in tool invocation. In this work, we introduce the concept of Personalized Tool Invocation and define two key tasks: Tool Preference and Profile-dependent Query. Tool Preference addresses user preferences when selecting among functionally similar tools, while Profile-dependent Query considers cases where a user query lacks certain tool parameters, requiring the model to infer them from the user profile. To tackle these challenges, we propose **PTool**, a data synthesis framework designed for personalized tool invocation. Additionally, we construct **PTBench**, the first benchmark for evaluating personalized tool invocation. We then fine-tune various open-source models, demonstrating the effectiveness of our framework and providing valuable insights.

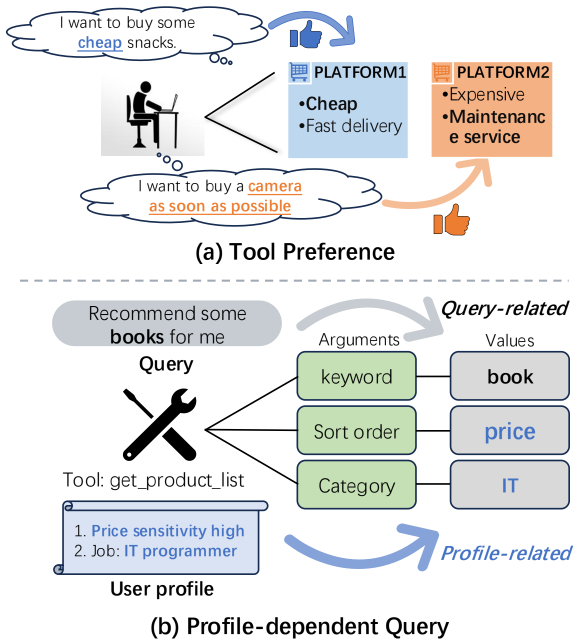


Figure 1: Example of Personalized Tool Invocation. (a) Tool Preference: Users may prefer different tools for similar functionalities depending on the query context. (b) Profile-dependent Query: Certain tool parameters may be missing from the user's query and need to be inferred from the user profile.

1 Introduction

Recently, large language models (LLMs) have demonstrated remarkable capabilities in natural language processing tasks, particularly in human-computer interaction, where they can effectively comprehend user queries and provide reasonable responses (Zhao et al., 2023). However, the knowledge embedded within LLMs is not inherently up-to-date, as updating these models requires extensive retraining with large-scale data, which incurs significant time and economic costs. To equip LLMs with the ability to solve complex problems and access the latest information, tool invocation capabilities are essential. For instance, LLMs can leverage mathematical tools to decompose and solve intricate mathematical problems or utilize internet APIs (Liu et al., 2025; Qin et al., 2024) and search engines (Schick et al., 2024; Nakano et al., 2021) to retrieve the most recent knowledge.

Existing research on enhancing LLMs's tool invocation abilities primarily focuses on improving fundamental capabilities (Qin et al., 2024; Yan et al., 2024; Lin et al., 2024), such as ensuring adherence to the required tool invocation syntax, comprehending tool functionalities, interpreting explicit user instructions, and extracting tool parameters. However, in real-world applications, user intents are often implicit rather than explicitly stated, requiring models to infer based on user-personalized profiles and behavioral history before invoking the appropriate tools. Two common scenarios illustrate this challenge on personalized tool invocation: **(1) Tool Preference**. When multiple tools offer similar functionalities, users often exhibit specific preferences. For example, in online shopping, users may choose different platforms depending on their preferences for particular product categories. Some users may prioritize platforms with superior maintenance services when purchasing high-value electronic products, despite the higher cost, while preferring platforms with faster delivery when buying inexpensive daily necessities. Inferring such preferences necessitates reasoning from user attributes, such as age, interests, and purchasing behavior. **(2) Profile-dependent Query**. In everyday scenarios, users tend to express their needs concisely and omit crucial details. For instance, a user might simply request, "Order me a hamburger from KFC," without specifying essential information such as the delivery address, recipient contact details, or preferred delivery time. This requires the model to infer the missing information based on the user profile, such as the user's work location, current time, and contact information, ensuring a seamless and accurate tool invocation process.

In this work, we propose the novel task of personalized tool invocation, aiming to address the aforementioned critical challenges. To enhance and systematically evaluate a model's ability in personalized tool invocation, we further introduce an automated data synthesis framework for this task, termed as **PTool**, which consists of three key stages: tool generation, user profile construction, and user behavior simulation. Firstly, we consider multiple commonly used real-world scenarios, where each scenario contains multiple functionally similar platforms organized in a hierarchical tree structure. We then leverage an advanced large language model (LLM) to recursively decompose platform functionalities using a depth-first expansion approach, progressively refining them until distinct tool APIs are defined for each functional category. This ensures that the generated tools comprehensively cover the functional demands of the given scenarios, thereby increasing the diversity of tools. Secondly, we abstract and summarize platform features and API parameters to extract both basic user attributes and personalized characteristics, including psychological traits and behavioral tendencies. To construct a diverse set of user profiles, we employ a bottom-up clustering approach for feature induction and a top-down assignment strategy for attribute allocation. Finally, we exploit the role-playing capabilities of LLMs to simulate user behaviors based on the assigned user profiles, generating both historical interactions and potential user queries. To establish reliable ground-truth labels, we further integrate a multi-agent framework that conditions query generation on user profiles. Following manual review and annotation, we construct **P**ersonalized **Tool****Bench** (**PTBench**), the first benchmark designed to evaluate large models' ability in personalized tool invocation, consisting of 1,083 high-quality annotated data samples. Our key contributions are summarized as follows:

- We propose the first paradigm for personalized tool invocation, incorporating both user tool preferences and profile-dependent user queries, two key challenges in real-world applications.
- We develop a systematic personalized data synthesis framework and construct PTBench, the first benchmark for personalized tool invocation, enabling a comprehensive evaluation of models' ability to invoke tools based on user information.
- We demonstrate that training open-source models on our synthesized dataset significantly improves personalized tool invocation capabilities, while also enhancing general tool invocation without compromising other general abilities.

2 Related Work

2.1 Tool Invocation

Tool invocation (also termed tool calling) involves tool selection from candidate tools and parameter extraction from queries. Existing works can be categorized into two tuning-free and tuning-based methods (Qu et al., 2025; Liu et al.). Tuning-free methods mainly rely on the prompt strategy with few-shot learning, involving encouraging LLM to reason by providing examples (Yao et al., 2022), rewriting tool documentation with LLMs to enhance the comprehension (Yuan et al., 2024), summarizing tool description with more concise and precise sentence (Xu et al., 2024), leveraging multi-agent collaboration to decompose the tool-calling task (Shi et al., 2024). Tuning-based methods leverage tool-learning samples to train existing LLMs, where the research problems comprise data collection and training strategy. Toolformer (Schick et al., 2024) and ToolkenGPT (Hao et al., 2024) add a special tool-related token into the vocabulary, switching the decoding process into tool selection and calling. Some works leverage advanced LLM to synthesize tool-calling samples to improve the tool-invocation ability of lightweight models, demonstrating the efficiency of the distillation from advanced models (Qin et al., 2024; Yang et al., 2023b; Liu et al., 2025).

2.2 Personalized LLMs

Personalized LLMs represent LLMs that have been adapted to align with user preferences and characteristics (Zhang et al., 2024c). Existing works mainly focus on the generation of personalized texts or applications in information systems. LLMs are customized as personal conversational AI assistants for various domains, including education (Kasneci et al., 2023; Dan et al., 2023; Park et al., 2024), healthcare (Belyaeva et al., 2023; Abbasian et al., 2024; Jin et al., 2024), finance (Liu et al., 2023; Lakkaraju et al., 2023), legal (Nguyen, 2023), and etc. User profiles are provided via prompts or hidden representation, leading the model to generate personalized text in the dialog. Personalized LLMs have been extensively applied in information systems such as recommender systems (Wu et al., 2023; Chen et al., 2024). LLMs are leveraged as an augmentation module for traditional recommender systems, serving as the content interpreter (Bao et al., 2023; Li et al., 2023; Yang et al., 2023a), the knowledge base (Xi et al., 2024; Wei et al., 2024), or the explainer (Lei et al., 2024; Wang et al., 2023). Also, many works directly deploy LLMs as the direct recommenders via prompt techniques (Lyu et al., 2024; Hou et al., 2024) or fine-tuning (Zhang et al.). However, there is no work considering personalization in tool learning. This work is the first to propose personalized tool invocation for LLMs.

3 Personalized Tool Invocation

We innovatively consider a practical and high-demand scenario in LLM tool invocation: **personalized tool invocation**. This scenario requires the model to leverage user-specific information when selecting and configuring tools to address user needs. In this chapter, we formally define the task of personalized tool invocation.

Given an LLM with model parameters θ, the general tool invocation task requires the model, when provided with a query q and a set of candidate tools T, to select the appropriate tool t^i and populate its corresponding parameters $a_1^i, \cdots, a_m^i$, forming the solution $A = [(t^i, a_1^i, \cdots, a_m^i), \cdots]$

In conventional formulations of this task, correctness is typically determined by whether the selected tool successfully resolves the query. However, this setting overlooks the fact that multiple tools may serve the same function (e.g., APIs from different platforms with similar capabilities), and that users often have preferences for certain tools—a concept we refer to as **tool preference**, defined as follows:

Definition 3.1. *(Tool Preference) User u prefers t^1 for query q_1 and t^2 for query q_2, where q_1, q_2 can be solved by both t^1 and t^2:*

$$t^1 \succ_{(u,q_1)} t^2; \quad t^2 \succ_{(u,q_2)} t^1 \tag{1}$$

Moreover, in A, both tool selection and parameter values are determined solely based on the information contained in the query. For instance, consider the query: "Book me a flight from Los Angeles to New York at 8:45 AM tomorrow". However, in real-world scenarios, users often do not provide such detailed query information. Instead, they may omit certain essential details required for tool invocation, meaning that the model cannot extract all necessary parameters from the query alone. We refer to this personalized scenario as an **profile-dependent query**, defined as follows:

Definition 3.2. *(Profile-dependent Query) Given the profile of the user u as P_u, the query q and the solution A, there exists value $\alpha \in A$, $\alpha \in P_u$ and $\alpha \notin q$, then the query q is called profile-dependent query.*

4 Personalized Tool Invocation Data Synthesis

To address the two challenges in personalized tool invocation mentioned above, we propose an automated data synthesis framework, PTool, for generating high-quality training and evaluation data for personalized tool invocation. The framework consists of three key stages: **Tool Generation**, **User Profile Construction**, and **Query and Solution Generation**, as illustrated in Figure 2. The detailed processes of each stage are described in the subsequent parts of this section.

4.1 Tool Generation

To cover the majority of scenarios encountered in daily life, we first constructed a diversified tool library across multiple contexts. Inspired by existing

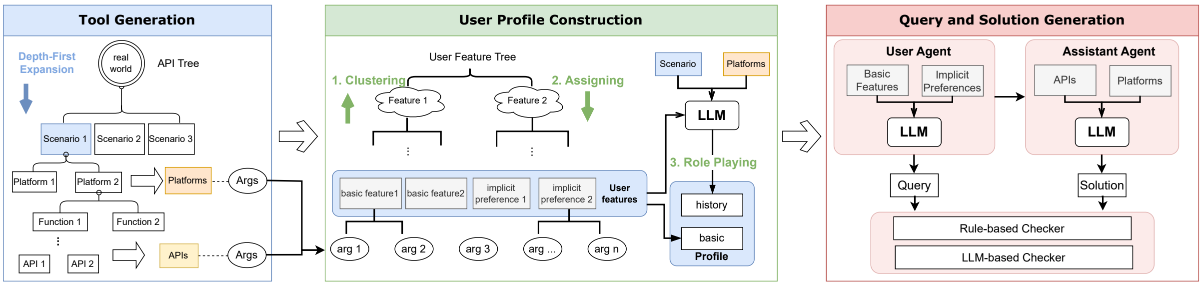


Figure 2: Framework of our personalized tool invocation data synthesis framework: PTool. The pipeline comprises three stages: Tool Generation, User Profile Generation and Query and Answer Generation.

work, we employed an advanced Large Language Model (LLM)-based data synthesis method to generate APIs. Similar to ToolACE, we also developed a structure akin to an API Tree, which allows for the generation of diverse tools.

Specifically, we initially define several demand scenarios from everyday life (e.g., shopping, food delivery, office) as the first-level nodes of the tree. Then, using a depth-first expansion approach, we iteratively refine the functionality at each node until we derive specific API descriptions as the leaf nodes. Notably, in order to generate data that enhances the model's Tool Preference capability, tools with similar functionalities are required. However, this API Tree expansion approach alone cannot achieve this. Therefore, at the second level of the tree expansion, we introduce the concept of platforms. For each scenario, we generated multiple platforms with distinct characteristics. For example, in the video entertainment scenario, platforms such as YouTube and TikTok were included, where YouTube focuses on long-form videos and TikTok emphasizes short, lifestyle-oriented clips. This enables us to obtain multiple tools with functionally interchangeable capabilities.

4.2 User Profile Construction

Personalization requires constructing diverse and realistic user profiles. This process involves three key challenges: (1) defining feature sets relevant to tool invocation, ensuring a structured linkage between user traits and tool selection; (2) maintaining sufficient diversity across profiles to enable generalization to unseen users; and (3) ensuring that profiles contain only observable basic and behavioral information, without incorporating detailed psychological attributes.

Bottom-up Feature Tree Construction. To systematically define user profile features, we adopt a tool-driven hierarchical clustering approach. We construct a feature tree, where platform characteristics and tool parameters serve as leaf nodes. Using advanced LLM-based clustering, we recursively merge semantically related parameters, summarizing them into higher-level features. This process continues until the number of parent nodes at each level falls within a predefined threshold. Notably, we categorize features during initial clustering: explicit basic features (e.g., age, gender) are directly observable, while implicit preferences (e.g., shopping preferences) remain latent and are used in subsequent user behavior generation.

Top-down Characteristic Assignment. Once the user feature tree is constructed, we encounter the second issue: how to diversify the assignment of values to these features to generate distinct user profiles. When using an advanced LLM to assign N different user features, two options typically arise: one is to assign all features for a single user at a time and repeat this process N times; the other is to assign all features for N users in one pass. The first method incurs higher inference costs and makes it challenging to avoid repetition across multiple generations, while the second is constrained by the model's context length limitation, especially when N or the number of features is large. Therefore, we adopt a top-down hierarchical assignment based on the tree structure. Specifically, for nodes at the l-th layer, we assign k_l different values simultaneously, and for the $(l+1)$-th layer nodes, the model generates k_{l+1} different values for each parent node's feature value. Thus, for a user feature tree with depth L, we can ultimately obtain $N = \prod_{l=0}^{L} k_l$ distinct user profiles. It's important to note that each time the LLM generates k_l, this number can be much smaller than N, allowing the LLM to generate diverse features in one pass.

User Behavior Generation. Once user profiles are assigned, they include both explicit basic features (e.g., occupation, gender, location) and implicit preferences (e.g., price sensitivity, product affinity). However, in real-world scenarios, user preferences are typically inferred through behavioral patterns rather than explicitly stated. To simulate authentic behavioral traits, we employ an LLM-based role-playing approach, where the model generates user actions on various platforms based on their profile and platform characteristics. For instance, given a user's preference for budget-conscious shopping, the model may generate interactions such as "searches for hiking backpacks on Amazon" or "purchases coffee from Walmart for $30." While implicit preferences remain unobservable to the model during task execution, they are embedded in prompts when generating tool invocation solutions, ensuring accurate and contextually appropriate tool selection.

4.3 Query and Solution Generation

For generating query-solution pairs, we adopt a multi-agent collaborative approach, involving two agents: the user agent and the assistant agent. The user agent generates queries by role-playing based on the user profile, while the assistant agent generates tool invocation solutions. The user agent's role information includes both basic and implicit features, as these provide a more accurate user representation than explicit behavioral features.

Given that a user's platform preferences may vary across queries, we explicitly incorporate platform information into the user agent's prompt. This enables the agent to generate queries aligned with the user's platform preferences. Additionally, we instruct the user agent to avoid revealing profile information in the queries, ensuring the generation of profile-dependent queries as well.

To ensure the correctness of tool invocations, we employ a two-tier verification strategy: rule-based validation and model-based verification. Rule-based validation checks the format of tool invocations to prevent issues such as unresolvable results or hallucinated tools and parameters. Model-based verification inputs the user profile, query, and solution triples into the LLM to verify parameter correctness, detect hallucinations, and assess whether the solution effectively resolves the query. Furthermore, to ensure evaluation accuracy, we manually inspect the correctness of tool invocation parameters. These parameters are annotated as profile-related or query-related, indicating whether they originate from the user profile or the query, facilitating more precise error feedback during evaluation.

Table 1: Statistics of our synthesized dataset. The samples in the test set are verified by human annotators. Trained and untrained represent the user profiles present and absent in the training set, respectively.

Dataset	#Scenario	#Platform	#API	#User	#Query
Train	5	15	360	74	7,096
Test(PTBench)	5	15	360	80	1,083
–Trained	5	15	360	74	474
–Untrained	5	15	360	6	609
Total	5	15	360	80	8,197

5 Experiments

5.1 Experimental Settings

Dataset Details. We leverage GPT-4-turbo to synthesize the personalized tool invocation dataset via our proposed framework. The overall dataset consists of a total of 80 users and 7,096 queries under 5 scenarios, including shopping, takeout, entertainment, work, and travel. Under each scenario, there are 3 platforms and 24 APIs in each platform as tools. We separate the dataset into training and test sets, randomly selecting all queries of 6 users and about 6% queries of another 74 users to form the test set PTBench. The 6 users will not be visible to models in the training process, termed as untrained. To ensure the quality of the test set, we manually verify each sample. The statistics are illustrated in Table 1.

Evaluation. We first evaluate the format accuracy by checking if the model's output can give formatted output, verifying the instruction following ability. The solution of each sample comprises two major parts: the platform and the tool invocation. The models are required to select the correct user-preferred platform and then generate suitable tool invocations. The platform accuracy demonstrates the ability of tool preference understanding. The tool invocation consists of three parts: tool name, parameters, and parameter values, where the parameter values comprise query-related and profile-related parameters. The profile-related parameters require the model to infer from the user profile, evaluating the ability to handle profile-dependent query. We calculate the accuracy of the function name, function parameter, and function value, respectively. The calculations of accuracy are detailed in Appendix A.1.

Baselines. We compare the latest open-source models and API-based models, as well as fine-tuned tool-calling models. Open-source models include

Table 2: Comparison with baseline models on PTBench. **Bold** and underline represent the best and the 2nd best results. **Preference** denotes the ability of tool preference. *T-** denotes the ability of filling correctness * in tool invocation. **DS-R1-Dis** is the abbreviation of DeepSeek-R1-Distill. All the results are accuracy.

Type	Model	Format	Preference	Param Value		Tool Invocation			Overall		
			Platform	*Query*	*Profile*	*T-name*	*T-param*	*T-value*	*Trained*	*Untrained*	*Overall*
API	**GPT-4-turbo**	**0.9778**	0.5484	**0.8123**	0.6832	0.9178	0.7709	**0.3518**	0.1834	0.1856	0.1847
	GPT-4o	0.9012	0.4484	0.7144	0.6104	0.8283	0.6991	0.2869	0.1350	0.1708	0.1551
	Deepseek-v3	0.9095	0.5280	0.7309	0.6416	0.8460	0.7530	0.3085	0.1708	0.1757	0.1736
	Deepseek-r1	0.8199	0.4819	0.6304	0.5806	0.7376	0.6294	0.2624	0.1477	0.1494	0.1486
	Qwen-max	0.7692	0.4946	0.6094	0.5440	0.7091	0.5843	0.2348	0.1456	0.1707	0.1597
	Claude-3.5-sonnet	0.9686	0.5826	0.7824	0.6504	0.7110	0.6445	0.2326	0.1329	0.1395	0.1367
OSS	**DS-R1-Dis-Llama-8B**	0.6427	0.3019	0.3823	0.3012	0.5080	0.3802	0.0981	0.0485	0.0394	0.0434
	DS-R1-Dis-Qwen-7B	0.6095	0.1469	0.2341	0.1039	0.3656	0.2113	0.0221	0.0042	0.0066	0.0055
	Qwen2.5-7B-Instruct	0.7858	0.3795	0.6132	0.4165	0.6833	0.5430	0.1837	0.0717	0.0755	0.0738
	Llama-3.1-8B-Instruct	0.8865	0.4053	0.6648	0.5141	0.7997	0.6252	0.2133	0.0929	0.0985	0.0960
	Mistral-7B-Instruct-v0.3	0.8587	0.3903	0.5598	0.3723	0.6612	0.3572	0.1450	0.0674	0.0559	0.0609
	Hammer2.1-7b	0.9649	0.3638	0.7296	0.5259	0.8402	0.6316	0.2262	0.0739	0.0689	0.0711
	ToolACE-8B	0.4035	0.1681	0.3289	0.2049	0.3887	0.2631	0.0906	0.0338	0.0378	0.0360
	Watt-tool-8B	0.3749	0.2281	0.2716	0.1990	0.3408	0.2218	0.0826	0.0591	0.0411	0.0489
	xLAM-7b-r	0.9529	0.3285	0.6794	0.4968	0.8688	0.5934	0.2217	0.0696	0.0771	0.0738
	Ours	0.9575	**0.7374**	0.7933	**0.7341**	**0.9242**	**0.8290**	0.3417	**0.2701**	**0.2660**	**0.2678**

DeepSeek-R1-Distill-Llama-8B(DeepSeek-AI, 2025), DeepSeek-R1-Distill-Qwen-7B(DeepSeek-AI, 2025), Qwen2.5-7B-Instruct(Team, 2024a,b), Llama-3.1-8B-Instruct (AI@Meta, 2024) and Mistral-7B-Instruct-v0.3(Jiang et al., 2023). API-based models include GPT-4-turbo[1], GPT-4o[1], Deepseek-v3(DeepSeek-AI, 2024), Deepseek-r1(DeepSeek-AI, 2025), Qwen-max(Team, 2024b) and Claude-3.5-sonnet[2]. Models fine-tuned for tool-calling include Hammer2.1-7b(Lin et al., 2024), ToolACE-8B(Liu et al., 2025), watt-tool-8B[3] and xLAM-7b-r(Zhang et al., 2024b; Liu et al., 2024; Zhang et al., 2024a).

Implementation Details. To validate the effectiveness of our model, we conducted various experiments by training LLMs with the synthesized dataset. We train the open-source LLM, Qwen2.5-7B-Instruct(Team, 2024a,b), in the supervised fine-tuning (SFT) manner. Due to limited resources, we adopt the parameter-efficient LoRA(Hu et al., 2022) training strategy to fine-tune the model. As for the hyper-parameters setting, we set the rank as 8, alpha as 16 learning rate as 10^{-4}, LR scheduler as cosine, WarmUp Ratio as 0.1 and epoch as 1 for all modules in the model.

[1]https://chatgpt.com
[2]https://www.anthropic.com
[3]https://ollama.com

5.2 Main Results

The overall results are illustrated in Table 2. The detailed results of trained and untrained users are presented in Appendix A.2. We have the following findings according to the results:

Finding 1: API-based large models significantly outperform smaller OSS models across various dimensions, including format compliance, tool preference capabilities, and tool invocation abilities. This aligns with the findings of most benchmarks, primarily attributed to the enhanced capabilities enabled by the larger scale of model parameters.

Finding 2: Most models fall short on the tool preference task, demonstrating low platform accuracy, including the state-of-the-art advanced model GPT-4-turbo. This phenomenon indicates that most LLMs fail to select suitable tools according to the user profile. Our model outperforms nearly all models in all aspects by a considerable improvement, presenting the necessity of personalized tool-invocation enhancement.

Finding 3: Our model demonstrates a significant improvement in its performance across various tasks on PTBench. Notably, the enhancement in the Tool Preference task is particularly pronounced when compared to the pre-trained Qwen2.5-7B-Instruct model. This also indicates that, even without additional manual verification of the training data, the model achieves a high accuracy, demonstrating the effectiveness of the proposed synthesis framework. Additionally, our model shows a significant improvement on untrained users, presenting the generalization of the model.

Finding 4: All models exhibit lower accuracy on profile-dependent parameter values compared to query-dependent parameters, indicating that inferring parameters from the profile presents a greater challenge. While our trained model does not surpass GPT-4-turbo in accuracy on query-dependent parameters, it outperforms larger models on profile-dependent parameters. Furthermore, the improvement over the pre-trained Qwen2.5-7B-Instruct model is more substantial, demonstrating the effectiveness of our data generation framework in handling the query-dependent query tasks.

5.3 Ablation Study

To investigate the importance of various parts in our synthesized user profile, we conduct the ablation study on the user profile, including 4 variants on the user profile:

- **All.** All information in the user profile is used, including basic features and behavioral history.
- **All w/o Basic.** Basic features are omitted.
- **All w/o History.** The behavioral history is given.
- **All w/o Basic&History.** Both basic features and behavioral history are omitted.

First, We use the four dataset variants to train and then evaluate the model with the consistent input. The results are reported in Table 3. From the result, we can observe that the existence of user history and basic features hold contributions to the overall performance of the model to an extent.

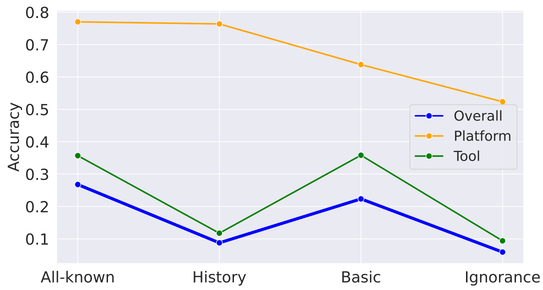
Additionally, we conduct experiments under two settings: (1) train the model with the All variant and evaluate the model with the four variants, illustrated in Figure 3a; (2) train the model with the four variants and evaluate the models with the All variant, illustrated in Figure 3b. The results exhibit that the model shows poor performance in the tool preference task when lacking user history information in training or evaluation. On the other hand, the accuracy of tool invocation suffers when basic features are absent, led by the challenging profile-dependent query task.

5.4 Error Analysis

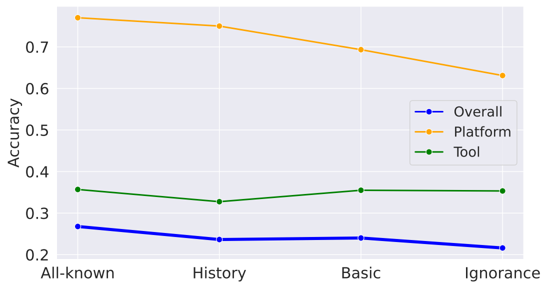
In the intention of gaining deeper insights into the function errors made by the models during the evaluation, we conduct investigations on the errors. We specifically choose our model, GPT-4-turbo and Qwen2.5-7B-Instruct to continue our investigation. We only analyze solutions with the correct format.

Table 3: Ablation of user profile on PTBench. The models are trained with various variants. The input in evaluation remains consistent with the training input.

Data	Untrained	Trained	Overall
All	**0.2660**	**0.2701**	**0.2678**
All w/o Basic	0.0969	0.2426	0.1606
All w/o History	0.2463	0.2531	0.2493
All w/o Basic&History	0.0591	0.0781	0.0674



(a) User profile ablation in evaluation.



(b) User profile ablation in training.

Figure 3: Ablation study on user profile in evaluation and training, respectively.

We analyze the function errors generally and divide them into 6 categories: wrong tools, missing tools, excessive tools, missing parameters, excessive parameters, and wrong parameters. The results are shown in Figure 4. From the pie chart, it is evident that filling the correct parameters is more challenging than the selection of the correct tools. After training with our synthesized data, the model is more familiar with the candidate tools, demonstrating less error percentage in tool selection.

5.5 Further Analysis

Model Scaling. For the purpose of analyzing the influence of model size on the performance of our trained model, we utilize models with different sizes in the Qwen2.5 series, including 7B, 3B, 1.5B and 0.5B. The results are shown in Figure 5. We

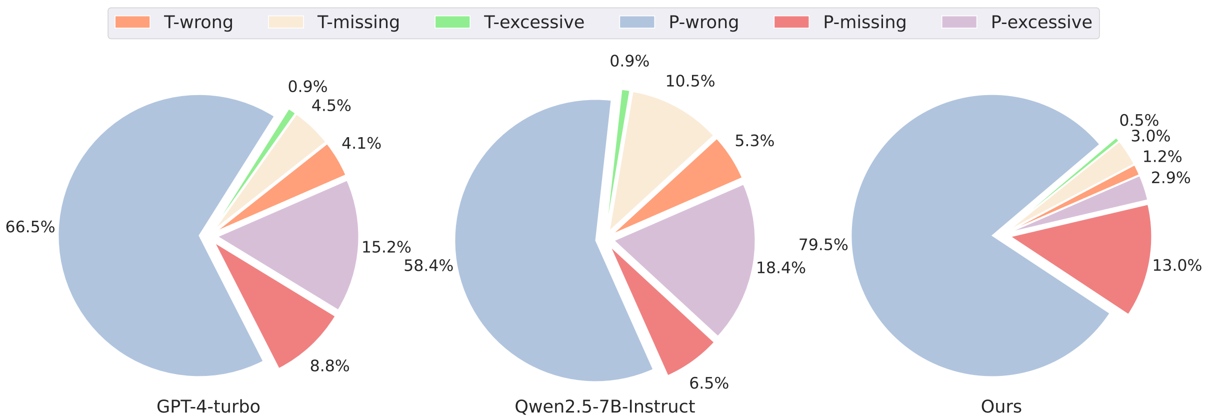


Figure 4: Error Analysis on PTBench. T-wrong, T-missing, and T-excessive represent wrong tools, missing tools and excessive tools, respectively. P-missing, P-excessive and P-error represent missing parameters, excessive parameters and wrong parameters, respectively.

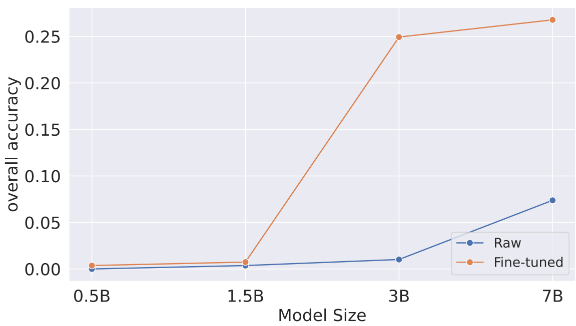


Figure 5: Study of model scaling. The base models are Qwen2.5-series.

can observe that the 1.5B and 0.5B model only show slight improvement from the training, while 3B and 7B model gain substantial improvement from the training. This demonstrate that the personalized tool invocation is a high-level capability of LLMs, requiring a certain scale of parameters.

General Capabilities. In order to validate that our synthesized data does not introduce negative effects on the model's general capabilities, we employ a diverse set of benchmarks to assess the performance from different perspectives, including general ability(MMLU(Hendrycks et al., 2021a,b)), coding(HumanEval(Chen et al., 2021)), math(GSM8K(Cobbe et al., 2021)), reasoning(CommonSenceQA(Talmor et al., 2019)) and basic function calling(tool-invocation) ability (BFCL non-live(Yan et al., 2024)). xLAM-7B-r, LLaMA-3-8B-Instruct, Raw Qwen2.5-7B-Instruct serve as baselines. The results are shown in Figure 6. From the figure, it is evident that there is no significance deterioration on abilities of our model compared to the raw model Qwen2.5-7B-Instruct. Nonetheless, our model gains a notable improvement on BFCL non-live, These findings suggest that our approach effectively enhances personalized functional calling capabilities without compromising the underlying LLM's other abilities.

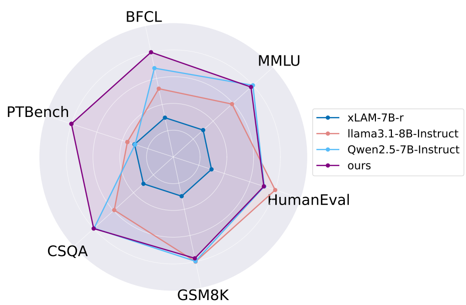


Figure 6: General Capabilities Analysis. Our model is fine-tuned from Qwen2.5-7B-Instruct.

6 Conclusion

In this work, we introduce the concept of personalized tool invocation, which encompasses two primary tasks: tool preference and profile-dependent queries. These tasks require the model's ability to understand the user's profile, select preferred tools based on historical behavior, and extract tool parameters from user information. To enhance and evaluate the model's personalized tool invocation capabilities, we propose a data synthesis framework and create a benchmark, PTBench, by manually inspecting a subset of the generated data. Extensive experimental evaluations assess the personalized tool invocation abilities of existing models, confirming the effectiveness of our synthesized data and its harmlessness to other model capabilities.

Limitations

We conclude the limitations of this work as follows:

First, the current coverage of scenarios is limited, as we primarily focus on the five most commonly encountered scenarios in daily life. However, this does not encompass the full spectrum of everyday needs. We plan to expand the range of scenarios covered by our tools in future work.

Second, personalized tool invocation is a crucial ability for LLMs in daily life. While we have proposed two key tasks in this work, they do not fully capture the entire scope of personalized tool invocation. One direction for future development is to introduce additional tasks that better address the diverse requirements of personalized tool invocation.

References

Mahyar Abbasian, Zhongqi Yang, Elahe Khatibi, Pengfei Zhang, Nitish Nagesh, Iman Azimi, Ramesh Jain, and Amir M Rahmani. 2024. Knowledge-infused llm-powered conversational health agent: A case study for diabetes patients. *arXiv preprint arXiv:2402.10153*.

AI@Meta. 2024. Llama 3 model card.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1007–1014.

Anastasiya Belyaeva, Justin Cosentino, Farhad Hormozdiari, Krish Eswaran, Shravya Shetty, Greg Corrado, Andrew Carroll, Cory Y McLean, and Nicholas A Furlotte. 2023. Multimodal llms for health grounded in individual-specific data. In *Workshop on Machine Learning for Multimodal Healthcare Data*, pages 86–102. Springer.

Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, et al. 2024. When large language models meet personalization: Perspectives of challenges and opportunities. *World Wide Web*, 27(4):42.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, and et al. 2021. Evaluating large language models trained on code.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yuhao Dan, Zhikai Lei, Yiyang Gu, Yong Li, Jianghao Yin, Jiaju Lin, Linhao Ye, Zhiyan Tie, Yougen Zhou, Yilei Wang, et al. 2023. Educhat: A large-scale language model-based chatbot system for intelligent education. *arXiv preprint arXiv:2308.02773*.

DeepSeek-AI. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2024. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems*, 36.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*, pages 364–381. Springer.

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*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Mingyu Jin, Qinkai Yu, Dong Shu, Chong Zhang, Lizhou Fan, Wenyue Hua, Suiyuan Zhu, Yanda Meng, Zhenting Wang, Mengnan Du, et al. 2024. Health-llm: Personalized retrieval-augmented disease prediction system. *arXiv preprint arXiv:2402.00746*.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274.

Kausik Lakkaraju, Sai Krishna Revanth Vuruma, Vishal Pallagani, Bharath Muppasani, and Biplav Srivastava. 2023. Can llms be good financial advisors?: An initial study in personal decision making for optimized outcomes. *arXiv preprint arXiv:2307.07422*.

Yuxuan Lei, Jianxun Lian, Jing Yao, Xu Huang, Defu Lian, and Xing Xie. 2024. Recexplainer: Aligning large language models for explaining recommendation models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1530–1541.

Ruyu Li, Wenhao Deng, Yu Cheng, Zheng Yuan, Jiaqi Zhang, and Fajie Yuan. 2023. Exploring the upper limits of text-based collaborative filtering using large language models: Discoveries and insights. *arXiv preprint arXiv:2305.11700*.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, Jun Wang, and Weinan Zhang. 2024. Hammer: Robust function-calling for on-device language models via function masking. *Preprint*, arXiv:2410.04587.

Weiwen Liu, Xingshan Zeng, Xu Huang, xinlong hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong WANG, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Wang Xinzhi, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. 2025. ToolACE: Enhancing function calling with accuracy, complexity, and diversity. In *The Thirteenth International Conference on Learning Representations*.

Xiao-Yang Liu, Guoxuan Wang, Hongyang Yang, and Daochen Zha. 2023. Fingpt: Democratizing internet-scale data for financial large language models. *arXiv preprint arXiv:2307.10485*.

Z Liu, Z Lai, Z Gao, E Cui, Z Li, X Zhu, L Lu, Q Chen, Y Qiao, J Dai, et al. Controlllm: augment language models with tools by searching on graphs (2023). *arXiv preprint arXiv:2310.17796*.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, et al. 2024. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *arXiv preprint arXiv:2406.18518*.

Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Chris Leung, Jiajie Tang, and Jiebo Luo. 2024. LLM-rec: Personalized recommendation via prompting large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 583–612, Mexico City, Mexico. Association for Computational Linguistics.

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*.

Ha-Thanh Nguyen. 2023. A brief report on lawgpt 1.0: A virtual legal assistant based on gpt-3. *arXiv preprint arXiv:2302.05729*.

Minju Park, Sojung Kim, Seunghyun Lee, Soonwoo Kwon, and Kyuseok Kim. 2024. Empowering personalized learning through a conversation-based tutoring system with student modeling. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–10.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36.

Zhengliang Shi, Shen Gao, Xiuyi Chen, Yue Feng, Lingyong Yan, Haibo Shi, Dawei Yin, Pengjie Ren, Suzan Verberne, and Zhaochun Ren. 2024. Learning to use tools via cooperative and interactive agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10642–10657, Miami, Florida, USA. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Qwen Team. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Qwen Team. 2024b. Qwen2.5: A party of foundation models.

Lei Wang, Songheng Zhang, Yun Wang, Ee-Peng Lim, and Yong Wang. 2023. LLM4Vis: Explainable visualization recommendation using ChatGPT. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 675–692, Singapore. Association for Computational Linguistics.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 806–815.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2023. A survey on large language models for recommendation. *CoRR*, abs/2305.19860.

Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Hong Zhu, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, and Yong Yu. 2024. Towards open-world recommendation with knowledge augmentation from large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 12–22.

Yang Xu, Yunlong Feng, Honglin Mu, Yutai Hou, Yitong Li, Xinghao Wang, Wanjun Zhong, Zhongyang Li, Dandan Tu, Qingfu Zhu, Min Zhang, and Wanxiang Che. 2024. Concise and precise context compression for tool-using language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16430–16441, Bangkok, Thailand. Association for Computational Linguistics.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Berkeley function calling leaderboard. `https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html`.

Fan Yang, Zheng Chen, Ziyan Jiang, Eunah Cho, Xiaojiang Huang, and Yanbin Lu. 2023a. Palr: Personalization aware llms for recommendation. *arXiv preprint arXiv:2305.07622*.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023b. GPT4tools: Teaching large language model to use tools via self-instruction. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Ren Kan, Dongsheng Li, and Deqing Yang. 2024. Easytool: Enhancing llm-based agents with concise tool instruction. *arXiv preprint arXiv:2401.06201*.

Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Juntao Tan, Thai Hoang, Liangwei Yang, Yihao Feng, Zuxin Liu, et al. 2024a. Agentohana: Design unified data and training pipeline for effective agent learning. *arXiv preprint arXiv:2402.15506*.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, et al. 2024b. xlam: A family of large action models to empower ai agent systems. *arXiv preprint arXiv:2409.03215*.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Xin Zhao, Leyu Lin, and Ji-Rong Wen. Recommendation as instruction following: A large language model empowered recommendation approach. *ACM Transactions on Information Systems*.

Zhehao Zhang, Ryan A Rossi, Branislav Kveton, Yijia Shao, Diyi Yang, Hamed Zamani, Franck Dernoncourt, Joe Barrow, Tong Yu, Sungchul Kim, et al. 2024c. Personalization of large language models: A survey. *arXiv preprint arXiv:2411.00027*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

A Experiments

A.1 Evaluation Metrics

The calculation of various metrics in PTBench are formulated as follows:

- **Format Accuracy** indicates the instruction-following ability.

$$format\_acc = \frac{\#parsable\ samples}{\#total} \tag{2}$$

- **Platform Accuracy** indicates the tool preference recognition ability.

$$platform\_acc = \frac{\#correct\ platform\ samples}{\#total} \tag{3}$$

- **Query-related Parameter-Value Accuracy** indicates the ability to extract values from query.

$$query\_param\_acc = \frac{\#correct\ query\ params}{\#total\ query\ params} \tag{4}$$

- **Profile-related Parameter-Value Accuracy** indicates the ability to extract values from profile.

$$profile\_param\_acc = \frac{\#correct\ profile\ params}{\#total\ profile\ params} \tag{5}$$

- **Tool Name Accuracy** indicates the tool selection ability.

$$tool\_name\_acc = \frac{\#correct\ name\ samples}{\#total} \tag{6}$$

- **Tool Parameter Accuracy** indicates the tool comprehension ability.

$$tool\_param\_acc = \frac{\#correct\ param\ samples}{\#total} \tag{7}$$

- **Tool Parameter-Value Accuracy** indicate the value extraction on context ability.

$$tool\_value\_acc = \frac{\#correct\,value\,samples}{\#total} \quad (8)$$

- **Overall Accuracy on Trained Users** indicate the personalized tool ability on trained users.

$$trained\_overall\_acc = \frac{\#correct\,trained\,samples}{\#trained\,total} \quad (9)$$

- **Overall Accuracy on Untrained Users** indicate the personalized tool selection ability on trained users.

$$untrained\_overall\_acc = \frac{\#correct\,untrained\,samples}{\#untrained\,total} \quad (10)$$

- **Overall Accuracy** indicate the overall personalized tool selection ability.

$$overall\_acc = \frac{\#correct\,samples}{\#total} \quad (11)$$

A.2 Detailed Results

The detailed results of the trained and untrained subset on PTBench are illustrated in Table 4 and Table 5, respectively.

B Examples

To enhance the understanding of the proposed personalized tool invocation, we illustrate an example in Figure 7.

Table 4: Comparison with baseline models on trained users in PTBench. **Bold** and underline represent the best and the 2nd best results.

Type	Model	Format	Preference	Param Value		Tool Invocation			Overall
			Platform	*Query*	*Profile*	*T-name*	*T-param*	*T-value*	
API	**GPT-4-turbo**	**0.9831**	0.5569	**0.7927**	0.7080	0.9325	0.7869	**0.3502**	0.1834
	GPT-4o	0.8840	0.4157	0.6520	0.6164	0.8143	0.6941	0.2637	0.1350
	Deepseek-v3	0.8903	0.5043	0.6868	0.6508	0.8376	0.7617	0.3059	0.1708
	Deepseek-r1	0.8376	0.4958	0.6112	0.6317	0.7637	0.6604	0.2574	0.1477
	Qwen-max	0.6941	0.4430	0.5083	0.5162	0.6393	0.5358	0.2152	0.1456
	Claude-3.5-sonnet	0.9662	0.5822	0.7519	0.6794	0.7152	0.6498	0.2236	0.1329
OSS	**DeepSeek-R1-Distill-Llama-8B**	0.6203	0.2891	0.3495	0.3111	0.4958	0.3925	0.1013	0.0485
	DeepSeek-R1-Distill-Qwen-7B	0.6013	0.1519	0.2148	0.0954	0.3503	0.1941	0.0147	0.0042
	Qwen2.5-7B-Instruct	0.7827	0.3882	0.5900	0.4447	0.6856	0.5612	0.1772	0.0717
	Llama-3.1-8B-Instruct	0.8819	0.3797	0.6384	0.5439	0.8039	0.6498	0.2236	0.0929
	Mistral-7B-Instruct-v0.3	0.8713	0.4198	0.5522	0.4113	0.6645	0.3734	0.1477	0.0674
	Hammer2.1-7b	0.9641	0.3650	0.7126	0.5468	0.8439	0.6582	0.2257	0.0739
	ToolACE-8B	0.4114	0.1709	0.3147	0.2061	0.3987	0.2721	0.0865	0.0338
	Watt-tool-8B	0.3966	0.2405	0.2708	0.2156	0.3586	0.2510	0.0992	0.0591
	xLAM-7b-r	0.9641	0.3586	0.6732	0.5315	0.8881	0.6329	0.2194	0.0696
	Ours	0.9662	**0.7826**	0.7791	**0.7653**	**0.9409**	**0.8628**	0.3333	**0.2701**

Table 5: Comparison with baseline models on untrained users in PTBench. **Bold** and underline represent the best and the 2nd best results.

Type	Model	Format	Preference	Param Value		Tool Invocation			Overall
			Platform	*Query*	*Profile*	*T-name*	*T-param*	*T-value*	
API	**GPT-4-turbo**	**0.9737**	0.5419	**0.8266**	0.6637	0.9064	0.7586	**0.3531**	0.1856
	GPT-4o	0.9146	0.4746	0.7596	0.6057	0.8391	0.7028	0.3054	0.1708
	Deepseek-v3	0.9245	0.5468	0.7629	0.6343	0.8522	0.7455	0.3104	0.1757
	Deepseek-r1	0.8062	0.4712	0.6443	0.5403	0.7175	0.6059	0.2660	0.1494
	Qwen-max	0.8276	0.5353	0.6828	0.5658	0.7635	0.6207	0.2496	0.1707
	Claude-3.5-sonnet	0.9704	0.5829	0.8046	0.6275	0.7077	0.6404	0.2397	0.1395
OSS	**DeepSeek-R1-Distill-Llama-8B**	0.6601	0.3120	0.4061	0.2935	0.5173	0.3695	0.0953	0.0394
	DeepSeek-R1-Distill-Qwen-7B	0.6158	0.1429	0.2481	0.1106	0.3777	0.2250	0.0279	0.0066
	Qwen2.5-7B-Instruct	0.7882	0.3727	0.6301	0.3943	0.6815	0.5287	0.1889	0.0755
	Llama-3.1-8B-Instruct	0.8900	0.4253	0.6839	0.4906	0.7964	0.6059	0.2052	0.0985
	Mistral-7B-Instruct-v0.3	0.8489	0.3678	0.5653	0.3416	0.6584	0.3448	0.1429	0.0559
	Hammer2.1-7b	0.9655	0.3629	0.7420	0.5094	0.8374	0.6109	0.2266	0.0689
	ToolACE-8B	0.3974	0.1659	0.3392	0.2039	0.3810	0.2562	0.0936	0.0378
	Watt-tool-8B	0.3580	0.2184	0.2722	0.1859	0.3268	0.2003	0.0706	0.0411
	xLAM-7b-r	0.9442	0.3054	0.6839	0.4695	0.8538	0.5632	0.2233	0.0771
	Ours	0.9507	**0.7028**	0.8035	**0.7096**	**0.9112**	**0.8030**	0.3481	**0.2660**

[SYSTEM]
You are given a user profile:

```
{
  "basic_features":{
        "username":"WineTraveler38",
        ...
  }
  "user_history":{
    "shopping":[{
            "platform":"MegaMart",
            "action":"Purchased a selection of premium imported wines"
        }
        ...
    ]
    ...
  }
}
```

Here is some platforms under the scenario:

```
[{
        "name":"MegaMart",
        "profile":{
            "product range":"A wide-ranging selection, offering products from various categories."
            ...
        }
    }
    ...
]
```

Here is some APIs under the platforms:

```
[{
        'type': 'function',
        'function': {
            'name': 'registerUser',
            'description': 'Registers a new user in the application.',
            'parameters': {
                'type': 'object',
                'properties': {
                    'username': {
                        'type': 'string',
                        'description': \"User's chosen username.\"
                    },
                    ...
                },
                'required': ['username', 'password', 'email']
            },
            'response': {
                'type': 'object',
                'properties': {
                    'success': {
                        'type': 'boolean',
                        'description': 'Status of registration.'
                    },
                    ...
                }
            }
        }
    },
    ...
]
```

The user will give you a query. Based on the profile, try to solve the query by using the platforms and APIs. The platform you choose should fit the user profile or the needs of the user's query. All the necessary information are provided in the user profile. DO NOT ask the user for further information. You should respond in the format of {platform:[func1(param1_name = param1_value, param2...), func2...]} No other text MUST be included.

[USER]
Could you please register an account for me using my username, password and email address, and setting my home location to my place of residence? I prefer not to receive any marketing emails.

[ASSISTANT]

```
{
  MegaMart:[
    registerUser(
      username='WineTraveler38', password='strongpassword123!',
      email='jeanlucbordeaux@email.com', preferredLanguage='French',
      marketingConsent=False, homeLocation='Paris, France'
    )
  ]
}
```

Figure 7: Illustration of personalized tool invocation task. The blue part represents the expected output.