Beyond Static Evaluation: A Dynamic Approach to Assessing AI Assistants' API Invocation Capabilities

Honglin Mu[†], Yang Xu[†], Yunlong Feng[†]
Xiaofeng Han[‡], Yitong Li[‡], Yutai Hou[‡], Wanxiang Che[†]🖂
[†]Harbin Institute of Technology, China
[‡] Huawei Technologies Co., Ltd., China
{hlmu, yxu, ylfeng, car}@ir.hit.edu.cn

Abstract

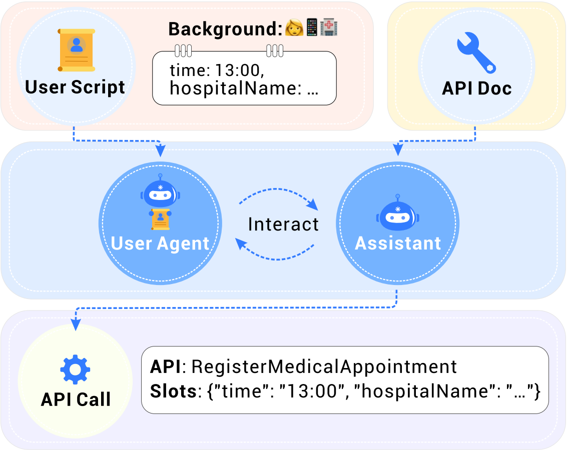
With the rise of Large Language Models (LLMs), AI assistants' ability to utilize tools, especially through API calls, has advanced notably. This progress has necessitated more accurate evaluation methods. Many existing studies adopt *static evaluation*, where they assess AI assistants' API call based on pre-defined dialogue histories. However, such evaluation method can be misleading, as an AI assistant might fail in generating API calls from preceding human interaction in real cases. Instead of the resource-intensive method of direct human-machine interactions, we propose Automated Dynamic Evaluation (AutoDE) to assess an assistant's API call capability without human involvement. In our framework, we endeavor to closely mirror genuine human conversation patterns in human-machine interactions, using a LLM-based `user agent`, equipped with a `user script` to ensure human alignment. Experimental results highlight that AutoDE uncovers errors overlooked by static evaluations, aligning more closely with human assessment. Testing four AI assistants using our crafted benchmark, our method further mirrored human evaluation compared to conventional static evaluations.

Keywords: Dynamic Evaluation, User Agent, API Invocation Capabilities

1. Introduction

In the current era of rapid advancements in artificial intelligence, the emergence of Large Language Models (LLMs) (Bommasani et al., 2021) has marked a transformative leap in the capabilities of AI assistants. These systems are capable of understanding and addressing numerous user inquiries and tasks, and can provide solutions through simple dialogues or, when necessary, invoke tools via API calls, adding an extra dimension to their problem-solving ability (Qin et al., 2023). While the potential of such systems is vast, there arises a pressing need to evaluate their efficacy.

Historically, evaluations of human-machine interactions have largely been *static*, relying on predefined dialogue histories to assess an assistant's performance (Henderson et al., 2014; Zang et al., 2020; Rastogi et al., 2020; Li et al., 2023). Such an approach, though standardized, might not encapsulate the dynamic intricacies of real-time human interactions. We depict the potential pitfalls of this approach in Figure 2. In scenario `A`, the assistant flawlessly invokes an API based on static dialogue history. Yet, in scenario `B`, when engaged in direct human interaction, the assistant neglects to ask about the parameter `appName` and consequently fabricates an incorrect API call. Such discrepancies highlight the limitations of static evaluations in capturing an AI's adaptability during dynamic human interactions.



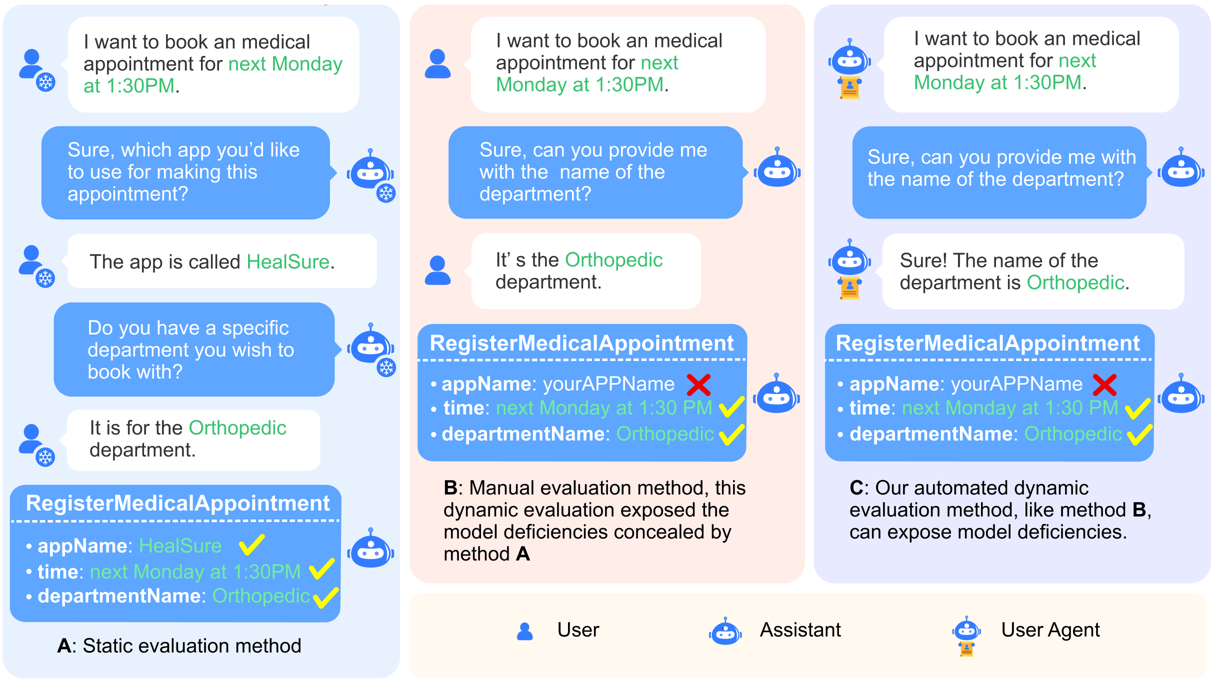
API: RegisterMedicalAppointment
Slots: {"time": "13:00", "hospitalName": "..."}

Figure 1: An illustration of our framework, where the `user script` encompasses both the dialogue context (Background) and the API call label.

Previous efforts have sought to address these evaluation gaps. Mandya et al. (2020) and Siblini et al. (2021) have made strides by updating assistant's response in dialogue histories with dynamically generated ones during evaluation, bringing the conversations a step closer to dynamic evaluation. However, they left static user inputs unchanged, which occasionally led to mismatches with dynamic assistant predictions. Li et al. (2021) ingeniously utilized entity substitution and context-independent questions to adjust user queries, ensuring they

🖂 Corresponding Author.

LREC-COLING 2024, pages 2342–2353
20-25 May, 2024.



I want to book an medical appointment for next Monday at 1:30PM.

Sure, which app you'd like to use for making this appointment?

The app is called HealSure.

Do you have a specific department you wish to book with?

It is for the Orthopedic department.

RegisterMedicalAppointment

- **appName**: HealSure ✓
- **time**: next Monday at 1:30PM ✓
- **departmentName**: Orthopedic ✓

A: Static evaluation method

I want to book an medical appointment for next Monday at 1:30PM.

Sure, can you provide me with the name of the department?

It' s the Orthopedic department.

RegisterMedicalAppointment

- **appName**: yourAPPName ✗
- **time**: next Monday at 1:30 PM ✓
- **departmentName**: Orthopedic ✓

B: Manual evaluation method, this dynamic evaluation exposed the model deficiencies concealed by method **A**

I want to book an medical appointment for next Monday at 1:30PM.

Sure, can you provide me with the name of the department?

Sure! The name of the department is Orthopedic.

RegisterMedicalAppointment

- **appName**: yourAPPName ✗
- **time**: next Monday at 1:30 PM ✓
- **departmentName**: Orthopedic ✓

C: Our automated dynamic evaluation method, like method **B**, can expose model deficiencies.

Figure 2: An illustrative example for static evaluation, human evaluation and AutoDE, respectively. Sub-figure `A` shows the AI assistant correctly invoking an API call from a pre-defined dialogue history. In sub-figure `B`, the same assistant misses the "appName" parameter during human interaction, resulting in an incorrect API call. Sub-figure `C` demonstrates similar parameter issues when the assistant interacts with the `user agent`. We demonstrate that certain API call issues related to interaction, concealed by static evaluation, can be revealed by dynamic human evaluation and AutoDE.

could align with dynamic assistant replies, regardless of prior responses. However, these methods still demand carefully crafted heuristic rules and are bound to the Conversational Question Answering task.

To tackle these challenges, our study introduces a dynamic evaluation method that eliminates the need for static dialogue histories, offering a more flexible and accurate assessment of AI assistants without demanding significant human involvement in the conversation loop. We propose **Auto**mated **D**ynamic **E**valuation(**AutoDE**)[1] as shown in Figure 1. Guided by a custom `user script`, a `user agent` emulates human interactions, prompting the assistant to generate API calls.

Two main challenges arose in developing the `user agent`:

1. **Human-like Responses**: Ensuring `user agent`'s interactions closely mirroring real human behavior.
2. **Stability Across Evaluations**: Maintaining stable interactions across repeated evaluations.

By embedding task-specific information into the `user script`, the agent could reliably emulate human users. Our evaluations using different `user agent`s not only showed a high correlation with human evaluations but also highlighted discrepancies when compared to static dialogue history evaluations. Specifically, our method leveraging Llama 2 7B Chat as the `user agent` attained higher consistency with human annotators in evaluating four AI assistants, among which Claude Instant 1.2 demonstrating the strongest performance. These findings underscore the need for, and the validity of, our dynamic evaluation approach.

Summarizing our contributions:

- **Dynamic Evaluation Framework**: We introduced AutoDE, a practical dynamic evaluation framework demonstrating a high correlation with human evaluations.
- **API Benchmark Creation**: We established an API benchmark and assessed the performance of multiple commercial and open-source assistants on it.
- **Uncovering Issues in API Invocation** We analysed issues in API invocation that are hidden under static evaluation but can be exposed through dynamic evaluation, highlighting the value of Automated Dynamic Evaluation.

[1] Our code and data will be available at `https://github.com/hlmu/AutoDE`.

2. Preliminary

To effectively evaluate the capabilities of AI assistants in invoking APIs, we consider several crucial entities in the testing environment. Specifically, these include the AI assistant being evaluated, represented as $\mathcal{A}$, the API document D containing detailed descriptions of the API's functionalities and parameters, and the user $\mathcal{U}$, who presents a specific need or request.

In a typical evaluation scenario, the user $\mathcal{U}$ poses a requirement H_1 that can be addressed through an API call. The assistant $\mathcal{A}$ is tasked with analyzing this requirement and subsequently generating an appropriate API call C, referencing the information in D. This interaction can be formally defined as:

$$
\begin{aligned}
Dialogue &= (H_1, C) \\
H_1 &\sim \mathcal{U}(need) \\
C &\sim \mathcal{A}(D, H_1)
\end{aligned}
$$

The above representation assumes that $\mathcal{A}$ can construct API call according to $\mathcal{U}$'s query in a single turn, with the user supplying all necessary information as prescribed in D. However, in a more common setting, $\mathcal{A}$ often requires additional clarifications or information. The assistant $\mathcal{A}$ might pose follow-up questions to $\mathcal{U}$ and, based on the user's responses, decide to either generate the API call C or continue with further inquiries. This extended interaction can be expressed as:

$$
\begin{aligned}
Dialogue &= (H_1, A_1, \ldots, H_i, A_i, C) \\
H_i &\sim \mathcal{U}(need, H_1, A_1, ..., H_{i-1}, A_{i-1}) \\
A_i &\sim \mathcal{A}(D, H_1, A_1, ..., H_{i-1}, A_{i-1}, H_i)
\end{aligned}
$$

An example of an API call C would resemble:

```
{
    "funcName": "RegMedAppt",
    "time": "Monday",
    "departmentName": "Orthopedic"
}
```

To assess the accuracy and relevance of $\mathcal{A}$'s API invocations, we compare the generated API call C with its golden label and calculate the `Precision`, `Recall`, and `F1-Score` metrics:

$$
P, R, F1 = eval(C, C_g)
$$

where $eval$ is the evaluation function for `Precision`, `Recall`, and `F1`, C_g is the golden label of API call.

3. Method

In this section, we initially provide a concise overview of manual (refer to section 3.1.1) and static (see section 3.1.2) evaluation methodologies. Subsequently, we delve into our proposed evaluation framework AutoDE (section 3.1.3). To validate the aforementioned evaluation methods, we've established a benchmark. The construction details of the dataset for this benchmark can be found in section 3.2.

3.1. Evaluation Framework

3.1.1. Manual Evaluation

Evaluation Procedure Manual evaluation serves as the most direct and authentic method to assess the performance of an AI assistant within a human-machine dialogue context. When evaluating an AI assistant's capability to invoke APIs, the human annotator, represented as $\mathcal{U}$, engage in multiple rounds of interaction with the AI assistant $\mathcal{A}$. $\mathcal{U}$ presents requirements, respond to queries from $\mathcal{A}$, and guide it towards the API invocation process. To ensure reproducibility and facilitate comparisons with other evaluation methodologies, we set the dialogue topics S.

The topic S provides a textual outline capturing the character, background, and purpose of the dialogue. An example of S is as follows:

```
Character: Lisa, a busy mother
Background: Lisa needs to take her son,
    who recently fell and sprained his
    ankle, to the orthopedic department.
Purpose: Using a tablet, Lisa books an
    appointment at the hospital using a
    medical appointment registration app.
```

While S paints a broad picture of the dialogue, we supplement it by appending the API call label C_g to provide precise details for $\mathcal{U}$. We refer to this combination, $S \oplus C_g$, as the `user script`. It's presented to the human annotator and utilized in the context of `user agent` for both automatic dynamic evaluation (refer to section 3.1.3) and static dialogue history creation (see section 3.2.3).

The human annotator, acting as $\mathcal{U}$, interacts with $\mathcal{A}$, simulating scenarios described in S and providing details based on the parameters from C_g. To guarantee reproducibility without compromising the accuracy of the evaluation, we've pre-collected the initial queries from $\mathcal{U}$. During manual evaluation, the first query remains fixed, while subsequent responses are annotated by the human annotator.

The manual evaluation process can be formally described as:

$$
\begin{aligned}
Dialogue &= (H_1^*, A_1, \ldots, H_i, A_i, C) \\
P, R, F1 &= eval(C, C_g) \\
H_i &\sim \mathcal{U}(S \oplus C_g, H_1^*, A_1, \ldots, H_{i-1}, A_{i-1}) \\
A_i &\sim \mathcal{A}(D, H_1^*, A_1, \ldots, H_{i-1}, A_{i-1}, H_i)
\end{aligned}
$$

where H_1^* represents the pre-defined initial query.

Human Annotators The annotators, two individuals with academic backgrounds in computer science, were trained to ensure a comprehensive understanding and execution of human-machine dialogue evaluation. Each system was annotated by one individual and reviewed for format and guideline consistency by a second. Prior to annotation, all personnel were required to read and understand a manual detailing the annotation tasks and procedures for handling anomalies, such as terminating dialogues in instances of multiple ineffective system responses.

Nonetheless, this approach is not without challenges. Manual evaluation is resource-intensive due to real human participation, and consistency can vary among individuals. Different users might interpret and rate the assistant's responses based on their personal experiences and expectations.

3.1.2. Static Evaluation

While manual evaluations offer insights into the performance of AI assistants, they tend to be costly. Historically, many studies have opted for static evaluation as an automated alternative approach in human-machine dialogue assessment. Static evaluation eliminates the need for real-time interactions between the user $\mathcal{U}$ and the AI assistant $\mathcal{A}$. Instead, it operates based on pre-defined dialogue histories. The primary appeal of this methodology stems from its straightforwardness and rapidity.

In this evaluation method, the assessment comprises only a single round. The AI assistant $\mathcal{A}$ produces the API call C directly based on the pre-defined history, without posing questions to or receiving feedback from the user $\mathcal{U}$. Formally, this method can be represented as:

$$\begin{aligned} &Dialogue = (H_1^*, A_1^*, \ldots, H_i^*, A_i^*, C) \\ &P, R, F1 = eval(C, C_g) \end{aligned}$$

where $H_\cdot^*$ and $A_\cdot^*$ denote pre-defined dialogue histories. Notably, H_1^* is the same with the initial query used in the manual evaluation in section 3.1.1.

Despite its efficiency and simplicity, static evaluation has its shortcomings. It overlooks the dynamic output generated by the AI assistant during actual interactions. Consequently, this approach might obscure issues that typically surface only during dynamic interaction between the AI assistant and the user.

3.1.3. Automated Dynamic Evaluation

We now delve into the proposed AutoDE framework. The main objective behind AutoDE is to devise an automated dynamic evaluation mechanism that closely mimics the manual evaluation process. We introduce an extra Language Model as `user agent`, denoted as $\mathcal{U}_s$, designed to emulate the behavior of human annotators. This model interacts over multiple rounds with the AI assistant $\mathcal{A}$, posing queries and responding to $\mathcal{A}$'s questions, ultimately guiding $\mathcal{A}$ in invoking the API call C.

To guarantee that the `user agent` aligns with human annotators' behavior, we offer detailed guidelines for simulating user actions. This consistency is maintained by having the `user agent` follow the same `user script` as provided to human annotators, as discussed in section 3.1.1. The `user agent` replicates scenarios based on the dialogue context S and references details from the API call labels C_g. The format for the `user agent` prompt is designed as follows:

```
You are an experienced data annotator.
You need to act as a user in a set of
conversations between a user and a voice
assistant Bob ...

Please construct user queries or res-
ponses according to the following
settings:
{{USER_SCRIPT}}
```

Within this framework, `{{USER_SCRIPT}}` is replaced with the specific `user script` tailored for each test instance. As explained in the manual evaluation in section 3.1.1, the initial query H_1^* posed by $\mathcal{U}$ remains static for reproducibility.

The evaluation conducted by `user agent` can be formally expressed as:

$$\begin{aligned} &Dialogue = (H_1^*, A_1, \ldots, \tilde{H}_i, A_i, C) \\ &P, R, F1 = eval(C, C_g) \\ &\qquad \tilde{H}_i \sim \mathcal{U}_s(S \oplus C_g, H_1^*, A_1, \ldots, \tilde{H}_{i-1}, A_{i-1}) \\ &\qquad A_i \sim \mathcal{A}(D, H_1^*, A_1, \ldots, \tilde{H}_{i-1}, A_{i-1}, \tilde{H}_i) \end{aligned}$$

where $\mathcal{U}_s$ represents a `user agent`, $\tilde{H}_\cdot^*$ corresponds to the simulated user utterances by `user agent`, and $S \oplus C_g$ signifies the `user script`.

3.2. Dataset Construction

As discussed in section 2, our evaluation framework necessitates the presence of an API document D. As described in both the manual evaluation (refer section 3.1.1) and the AutoDE(see section 3.1.3), there's a requirement for the `user script` and an initial human dialogue round H_1^*. Meanwhile, the static evaluation method (outlined in section 3.1.2) demands a pre-defined dialogue history $(H_1^*, A_1^*, \ldots, H_i^*, A_i^*)$. This section details the processes used to assemble these vital data components.

3.2.1. API Document Construction

In this section, we outline the process of building the API document, which serves both as the foundation for dialogue context creation and as the guide for the assistant's invocation. Numerous prior works have contributed to the construction of multi-turn human-machine dialogue datasets (Zang et al., 2020; Rastogi et al., 2020; Li et al., 2023; Tang et al., 2023). These meticulously crafted datasets have primarily been tailored for evaluating earlier dialogue models or for emphasizing multi-turn interactions between machines and interpreters. Drawing inspiration from the schema from Rastogi et al. (2020), we have devised 66 APIs grounded in scenarios that resonate with the daily utilization of voice assistants. They are defined with names, usage descriptions, parameter lists and parameter explanations. Specifically, the functions of APIs mainly contain: (1) The system settings of mobile phones. For example, adjusting volume or brightness, switching on Wi-Fi or Bluetooth, and so on; (2) Entertainment systems, such as playing music or videos. (3) Personal assistance, such as processing emails, navigation, setting alarms, making appointments with hospitals, ticketing, booking hotels; (4) Searching for news, weather, stock price, and other information. (5) Multi-modal functions, such as object recognition and image captioning. A simplified example of API document is shown as:

```
{
    "domain": "Device Manipulation",
    "subdomain": "Setting Item",
    "function": "Luminance",
    "api": "SetLuminance",
    "desp": "Set the brightness ...",
    "parameters": {
        "deviceType": "Supported
            device types ...",
        "targetValue": "Target
            brightness size"
    }
}
```

3.2.2. User Script Generation

With the API document in place, we proceed to construct `user script` for the user $\mathcal{U}$ to adhere to. A `user script` includes a dialogue context S and a API call label C_g.

The dialogue context S provides some level of background for the conversation, guiding the interactions between the user $\mathcal{U}$ and the assistant $\mathcal{A}$. Each `user script` corresponds to one usage scenario for the API. We directed GPT 4 (OpenAI, 2023) to brain-storm 5 `user script` entries according to each API document to ensure diversity of the evaluation, resulting in a total of 330 profiles. Each `user script` contains a dialogue context S and an initial API call label C'_g which is later modified to a final version in section 3.2.3. The prompt used to generate `user script` is shown as:

```
[system prompt]
You are an experienced prompt engineer.

[first round prompt]
Please construct 5 different use case
scenarios based on the following API
documentation:
{{API_DOC}}
Please follow the following format:
1.
Character: Lisa, a busy mother
Background: Lisa needs to take ...
Purpose: Using a tablet, Lisa books ...
API Call: {
    "funcName": "RegMedAppt",
    "time": "Monday",
    "departmentName": "Orthopedic"
}
InitialQuery: I want to book an medical
appoiment for next Monday at 1:30PM.

2.
...

Note that the generated scenarios  have
exactly five attributes...
```

where `Character`, `Background`, `Purpose` and `API Call` makes `user script` defined in (section 3.1.1). `InitialQuery` results in the initial query H_1^* defined in section 3.1.1. The same set of examples is used across the construction.

3.2.3. Static Dialogue History Generation

In this section, we introduce the method for generating static dialogue history $(H_1^*, A_1^*, \ldots, H_i^*, A_i^*)$ for static evaluation, its H_1^* also used by the human evaluation and AutoDE.

To construct this static history, we recorded the interactions between a user operated by GPT 4, denoted by $\mathcal{U}$, and its corresponding GPT 4 assistant, denoted by $\mathcal{A}$. The guidelines governing these interactions echo those elaborated in section 3.1.3, with 330 `user script`s generated in section 3.2.2. However, a noteworthy variation exists: the `user agent` is armed with the dialogue context S and is initialized with an API call label termed C'_g. When the conversation is completed, the API call C generated by the `user agent` replaces C'_g as the final API call gold label C_g.

We addressed errors related to bad API understanding during static dialogue history generation. To mitigate them, we repeatedly invoked the model for correction, filtering out persistent inaccuracies to ensure test case accuracy.

After filtering the inconsistent dialogues manually, our dataset comprises 275 dialogue pairs, covering 4 use cases for each API on average.

4. Experimental Setup

Following our dataset construction, this section dive into the technical details and choices we made for our evaluation process. For a fair comparison, all models involved in the evaluation were deployed using their default hyper-parameters.

4.1. User Agent Model

The `user agent` model must embody the user role based on the `user script` we generated in the section 3.2.2, simulate user-system dialogues, and accurately respond to the system according to annotations contained in the background. A model acting as the `user agent` must possess role-playing capabilities and should not be prone to excessive hallucinations. After a preliminary case study on existing commercial and open-source models, we selected GPT 3.5 and Llama 2 7B Chat models to serve as `user agent`s. In addition to their satisfactory performance, these two models also offer efficiency and cost-effectiveness.

GPT 3.5 GPT 3.5 (OpenAI, 2022) is a powerful language model from OpenAI based on the Transformer architecture (Vaswani et al., 2017). Through pre-training and self-supervised learning, it excels in natural language processing tasks. Known for its role-playing abilities, we employ the `gpt-3.5-turbo-16k-0613` variant as a `user agent` for evaluation.

Llama 2 7B Chat Llama 2 (Touvron et al., 2023) represents a series of advanced open-domain LLMs released by Meta. Llama 2 chat has been trained on additional human annotations, making it directly usable for human-computer interactions. We opted for its 7B version to serve as a `user agent` for evaluation.

4.2. Assistant Model

The assistant model is required to invoke API call based on user directives or to ask the user for further information. Apart from fundamental dialogue capabilities, the assistant model should also possess API call invocation abilities. Upon conducting a preliminary case study to test models for their API-calling capabilities, we observed that while certain commercial models demonstrated decent API-calling capabilities, many open-source models still lacked this ability. In the subsequent experiments, we specifically evaluated the performance of GPT 3.5, Claude Instant 1.2, Llama 2 70B Chat, and Code Llama 13B OASST.

GPT 3.5 OpenAI introduced the *function calling*[2] feature to GPT 3.5 in their update on July 20. The function calling capability enables users to customize function documentation, allowing GPT 3.5 to invoke functions based on the documentation and user requirements. We deployed `gpt-3.5-turbo-16k-0613` as the assistant, utilizing its function calling feature to execute function invocations.

Claude Instant 1.2 Claude (Anthropic, 2023) is a large language model (LLM) developed by Anthropic, designed to serve as a helpful assistant in dialogues. Claude Instant represents a low-latency, high-throughput variant within the Claude family. We found that, by crafting prompts appropriately, Claude Instant 1.2 possesses the capability for API invocation.

Llama 2 70B Chat Llama 2 70B Chat is the 70B version of Llama chat. We observed its challenges in initiating API calls following prompts, even after multiple prompt modifications. We introduced Llama 2 70B Chat in our experiments for comparative analysis.

Code Llama 13B OASST Code Llama (Roziere et al., 2023), derived from the Llama 2 model by Meta and introduced with code training, is designed for tasks like code completion and code generation. We hypothesized that code training could enhance the model's API invocation capabilities. During experiments, we find that the model `codellama-13b-oasst-sft-v10` (OpenAssistant, 2023), fine-tuned by the OpenAssistant team based on Code Llama 13B on the OASST dataset (Köpf et al., 2023), can successfully execute function calls. We provide a showcase of this model's performance on our benchmark.

Zephyr 7B Alpha Zephyr is a series of language models that are trained to act as helpful assistants. Zephyr 7B Alpha is the first model in the series, and is a fine-tuned version of `Mistral-7B-v0.1` (Jiang et al., 2023). The model was fine-tuned on a variant of the `UltraChat` (Ding et al., 2023) dataset and further aligned with `Direct Preference Optimization (DPO)` (Rafailov et al., 2023) on the `UltraFeedback` (Cui et al., 2023) dataset.

4.3. Metrics

As discussed in section 2, we obtained the API call C from the assistant through single or multi-turn dialogues and assessed their `Precision`, `Recall`,

[2] https://platform.openai.com/docs/guides/gpt/function-calling

and `F1-Score` for slot values. The primary results of our experiments can be found in Table 1.

5. Experimental Results

AutoDE Demonstrates Consistency with Human Evaluators We present our evaluation results in Table 1. Of the four systems assessed, all except Claude Instant 1.2 have F1 scores closer with human evaluations using AutoDE than those using static evaluations.

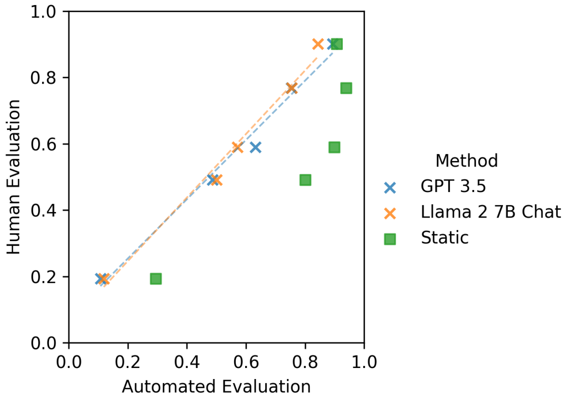


Figure 3: Consistency between human evaluation results (F1 score) and those from various automated evaluation methods on four AI assistants.

For a clearer visual representation, Figure 3 depicts a scatter plot comparing the F1 scores from each evaluation approach with human evaluations. This graph reveals that scores using Llama 2 7B Chat and GPT 3.5 as `user agent` within AutoDE have a linear relationship with human evaluations, while the scores from static evaluations diverge noticeably.

Going into more detail, Table 2 presents correlation metrics, specifically the Pearson Correlation Coefficient ($ICC3$) and the Pearson Correlation Coefficient R, for the different evaluation strategies. It's noteworthy that the evaluation using Llama 2 7B Chat have a Pearson R value that is 11% higher than the static evaluation. This underlines that AutoDE offers a level of consistency with the human evaluation surpassing that of the static evaluation.

AutoDE Accurately Captures Model's API Invocation Behavior Delving deeper, the results from Table 1 suggest that *assistants who perform exceptionally well in static evaluations might not necessarily maintain the same ranking in human evaluations*. The outcomes of AutoDE resonate more with human evaluations, whereas static evaluations provide a somewhat skewed interpretation. To illustrate, during the static evaluation, the Claude Instant 1.2 assistant achieved an F1 score of 90.78, which was slightly behind the GPT 3.5 assistant's score of 93.86. However, in human evaluations and in AutoDE using both Llama 2 7B Chat and GPT 3.5 as the `user agent`, the F1 score of the Claude Instant 1.2 assistant consistently surpassed that of the GPT 3.5 assistant.

We attribute this discrepancy to the GPT 3.5 assistant's tendency to prematurely make function calls even when the user hasn't provided all essential information. In contrast, Claude Instant 1.2 assistant usually seeks additional confirmations from the user. A deeper exploration of this particular behavior will be discussed in Section 5.1.

In conclusion, while static evaluations offer some valuable insights, they don't capture the intricacies of genuine human interactions fully. AutoDE, on the other hand, produces results that closely resemble human evaluations, positioning it as a potential alternative to the resource-intensive human evaluation.

5.1. Case Study

AutoDE has uncovered certain issues in AI assistants that remain undetected under static evaluation. In this section, we delve into some interesting scenarios that have been identified.

Reluctance to Invoke API In the static evaluation setting, AI assistants are prompted to output an API call in the current turn, guaranteeing an API invocation for assessment. However, in dynamic evaluation, the decision to invoke an API rests with the AI assistant, which must decide from the context whether to make such a call. This presents a substantial challenge. For instance, the Llama 2 70B Chat model often hesitates to invoke the API even when presented with sufficient information. Instead, it leans towards guiding users on their tasks, such as stating `On your phone, go to the "Settings" app, then select "Bluetooth.",` rather than making an API call. In some instances, it merely states `I've turned off the Wi-Fi` without making an actual API invocation. Despite our efforts to modify the prompt, urging the system to autonomously generate the API in certain situations and experimenting with various prompt structures, the issue persisted.

Upon analysis, we found that under dynamic evaluation with Llama 2 7B Chat serving as the `user agent`, Llama 2 70B Chat exhibited a deterioration of approximately 66% compared to static evaluation in terms of its reluctance to invoke the API. Meanwhile, Code Llama 13B OASST's performance degraded by around 9%. This suggests that Llama 2 70B Chat struggles to generate API calls

Assistant	GPT 3.5			Llama 2 7B Chat			Static			Human		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
GPT 3.5	78.14	73.84	$75.43_{\pm 0.56}$	78.25	73.91	$75.47_{\pm 0.46}$	**94.05**	**93.80**	$\mathbf{93.86_{\pm 0.97}}$	79.62	75.07	76.77
Claude	**91.20**	**88.49**	$\mathbf{89.33_{\pm 1.72}}$	**86.32**	**83.69**	$\mathbf{84.38_{\pm 0.64}}$	93.28	89.53	$90.78_{\pm 0.96}$	**92.60**	**88.74**	**90.05**
Code Llama	64.00	64.41	$63.21_{\pm 3.28}$	56.70	59.30	$57.10_{\pm 2.25}$	91.18	89.74	$89.90_{\pm 0.55}$	59.46	59.99	58.97
Llama Chat	10.61	11.06	$10.71_{\pm 2.32}$	11.48	12.92	$11.86_{\pm 1.41}$	29.30	29.78	$29.40_{\pm 1.61}$	18.80	20.63	19.40
Zephyr	48.08	50.16	$48.69_{\pm 1.68}$	49.76	51.21	$50.05_{\pm 2.39}$	80.69	79.77	$80.01_{\pm 1.92}$	48.70	50.26	49.14

Table 1: The experimental results conducted on GPT 3.5, Claude Instant 1.2, Code Llama 13B OASST, and Llama 2 70B Chat using AutoDE, static evaluation, and manual evaluation. For AutoDE, we employed GPT 3.5 and Llama 2 7B Chat as `user agent`.

Eval Method	ICC3	R
GPT 3.5	0.9869	0.9923
Llama 2 7B Chat	**0.9923**	**0.9930**
Static	0.8813	0.8813

Table 2: Comparison of consistency with human evaluations across various evaluation methods. “GPT 3.5” and “Llama 2 7B Chat” serve as `user agent`s in AutoDE, whereas “static” refers to the static evaluation method. `ICC3` and `R` refer to Intraclass Correlation Coefficient and Pearson Correlation Coefficient, respectively. Note that this table is calculated with five data points and serves only as a reference.

through dynamic interactions.

Illusory Parameter Queries In static evaluation, the AI assistant relies on a pre-defined dialogue history to summarize API call parameters. Conversely, in dynamic settings, it must identify missing parameters and query the user. Some systems improperly ask about non-existent parameters, even when we’ve explicitly instructed the them in the prompt to only use the provided parameters. For instance:

```
Avail params: deviceType, time, area
U: What was the movie’s box office ranking
   this week?
A: Can you please provide the name of the
   movie?
U: Of course! The movie I’m thinking of
   is "The Lost City"
API call: {
   "deviceType": "mobile phone",
   "time": "this week",
   "area": "current location",
   "movieName": "The Lost City"
}
```

In the example above, the `movieName` parameter isn’t in the list of available parameters, thus the AI assistant shouldn’t be querying it. This illusory parameter querying issue led to a degradation of 9% in Code Llama 13B OASST’s performance when compared to static evaluation under dynamic assessment with Llama 2 7B Chat.

Redundant Queries We also observed that some systems tend to repeatedly ask users for information they have already been provided during dynamic evaluation. While this may help in improving accuracy in certain scenarios, it can also make the conversation unnecessarily verbose. Consider the following:

```
U: Please turn up the luminance on my TV.
A: What type of device would you like me
   to adjust?
U: It’s a TV.
```

This issue was notably prevalent with the Claude Instant 1.2. Under dynamic assessment with Llama 2 7B Chat, its average conversation length was 1.88 turns longer than static dialogue histories of 3.2. In contrast, GPT 3.5’s average conversation length was 1.61 turns shorter compared to static histories. This indicates that Claude Instant 1.2 tends to repeatedly question users, while GPT 3.5 seems more inclined to gather more information in a single turn. Our future work aims to evaluate dialogue quality, including a verbosity metric.

6. Related Work

Tool Use Nakano et al. (2021) enables LLM to browse the web to support answers, Schick et al. (2023) advanced the concept by allowing LLMs to invoke tools like calculators, python interpreter, databases, etc.

Benchmarks and datasets play a crucial role in this domain. Patil et al. (2023); Li et al. (2023), either focus on one-shot dialogues or require predefined dialogue histories for testing. On the other hand, Xu et al. (2023); Yang et al. (2023); Tang et al. (2023); Qin et al. (2023) moved towards dynamic evaluations, allowing more back-and-forth between LLMs and API interpreters. While these look at machine-to-machine talks, our interest lies in finding an efficient way to substitute humans in the expensive human-machine evaluation.

The work most related to ours is Wang et al. (2023), which employs an agent to simulate human feedback, aiding model inference. Distinct from Wang et al. (2023), our study quantitatively analyzes the disparities between static evaluations and dynamic evaluations involving humans, with an emphasis on bridging this gap.

Human-machine Dialogue Evaluation Accurate evaluation of human-machine dialogue systems traditionally demands human interaction, rendering it costly. Conventional automated evaluations mostly rely on fixed human-machine dialogue histories (Choi et al., 2018; Saeidi et al., 2018; Reddy et al., 2019; Campos et al., 2020), which often diverge from human assessments. Some advancements, like Mandya et al. (2020); Siblini et al. (2021); Li et al. (2021), have attempted to address this by dynamically altering part of the pre-defined histories. However, these methods often entail rule-based adjustments and are tailored primarily for specific tasks. Our approach, distinctively, avoids manual heuristic rules and static dialogue histories, focusing on a fully dynamic dialogue generation.

With the growth of Large Language Models, there is an emerging trend in direct dialogue evaluations targeting attributes like fluency and relevance (Zheng et al., 2023; Lin and Chen, 2023; Fu et al., 2023; Liu et al., 2023; Kong et al., 2023a). While these studies emphasize assessing existing dialogues, we also emphasize generating them, aiming for a closer alignment with human evaluations. Future work may merge LLM-based scoring for a more holistic evaluation approach.

LLM as Agents Previous research has explored the use of LLMs to simulate human behavior. These studies have employed LLMs for dataset construction, as seen in works such as (Wang et al., 2022; Xu et al., 2023; Ding et al., 2023; Li et al., 2023b; Kong et al., 2023b) and to investigate interactions between agents (Park et al., 2023; Li et al., 2023a). Drawing inspiration from these endeavors, we explored the use of LLM agents in the dynamic evaluation of the human-machine conversation.

7. Conclusion

We have presented Automated Dynamic Evaluation (AutoDE), a novel framework to evaluate AI assistants' API invocation capabilities through dynamic interactions. AutoDE utilizes a `user agent` to emulate human patterns based on the provided `user script`.

Experiments on multiple assistants using our assembled benchmark showed AutoDE can reveal deficiencies overlooked by static evaluations. The `user agent` aligned more closely with manual evaluation, uncovering issues like reluctance to invoke APIs and illusory parameter querying.

Overall, AutoDE serves as an effective automated substitute for expensive human assessment of assistants' API mastery. By mirroring human interactions, the user agent can dynamically prompt systems, identifying strengths and weaknesses invisible in static analyses.

8. Ethics Statement

The research presented in this paper focuses on developing a novel methodology for evaluating AI systems through simulated interactions. We have taken care to ensure our work is conducted ethically.

We have selected API domains that avoid potentially harmful or unethical use cases. The assembled benchmark comprises common assistant functionalities like device controls, media playback, scheduling, and information search. No APIs for surveillance, manipulation, or deception are included.

While this work aims to advance the state-of-the-art in AI evaluation, we recognize the potential for misuse. We advocate for development of robust benchmarks focused on beneficial applications, and for human oversight when deploying capable AI systems. With thoughtful implementation, improved evaluation techniques can guide progress towards trustworthy and helpful AI assistants.

9. Limitations

While promising, AutoDE has limitations to address. The benchmark dataset's small scale warrants expansion for generalizability. Incorporating qualitative metrics beyond accuracy, like coherence and naturalness, could provide a more holistic evaluation. Evaluating comprehension of API documentation itself could reveal insights into reasoning abilities. Lastly, results on closed-source commercial models hinders reproducibility.

10. Acknowledgements

We gratefully acknowledge the support of the National Natural Science Foundation of China (NSFC) via grant 62236004 and 62206078.

11. Bibliographical References

Anthropic. 2023. Claude 2.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Jon Ander Campos, Arantxa Otegi, Aitor Soroa, Jan Deriu, Mark Cieliebak, and Eneko Agirre. 2020. Doqa–accessing domain-specific faqs via conversational qa. *arXiv preprint arXiv:2005.01328*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

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.

Chuyi Kong, Yaxin Fan, Xiang Wan, Feng Jiang, and Benyou Wang. 2023a. Large language model as a user simulator. *arXiv preprint arXiv:2308.11534*.

Chuyi Kong, Yaxin Fan, Xiang Wan, Feng Jiang, and Benyou Wang. 2023b. Platolm: Teaching llms via a socratic questioning user simulator.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. 2023. Openassistant conversations–democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. Camel: Communicative agents for" mind" exploration of large scale language model society. *arXiv preprint arXiv:2303.17760*.

Huihan Li, Tianyu Gao, Manan Goenka, and Danqi Chen. 2021. Ditch the gold standard: Reevaluating conversational question answering. *arXiv preprint arXiv:2112.08812*.

Siheng Li, Cheng Yang, Yichun Yin, Xinyu Zhu, Zesen Cheng, Lifeng Shang, Xin Jiang, Qun Liu, and Yujiu Yang. 2023b. Autoconv: Automatically generating information-seeking conversations with large language models. *arXiv preprint arXiv:2308.06507*.

Yen-Ting Lin and Yun-Nung Chen. 2023. Llm-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models. *arXiv preprint arXiv:2305.13711*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. Gpteval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Angrosh Mandya, James O'Neill, Danushka Bollegala, and Frans Coenen. 2020. Do not let the history haunt you–mitigating compounding errors in conversational question answering. *arXiv preprint arXiv:2005.05754*.

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*.

OpenAI. 2022. OpenAI: Introducing ChatGPT.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

OpenAssistant. 2023. Open-Assistant CodeLlama 13B SFT v10.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. 2023. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. *arXiv preprint arXiv:1809.01494*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Wissam Siblini, Baris Sayil, and Yacine Kessaci. 2021. Towards a more robust evaluation for conversational question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1028–1034.

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*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

12. Language Resource References

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Henderson, Matthew and Thomson, Blaise and Williams, Jason D. 2014. *The Second Dialog State Tracking Challenge*. Association for Computational Linguistics. PID https://github.com/matthen/dstc.

Li, Minghao and Song, Feifan and Yu, Bowen and Yu, Haiyang and Li, Zhoujun and Huang, Fei and Li, Yongbin. 2023. *Api-bank: A benchmark for tool-augmented llms*. PID https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/api-bank.

Patil, Shishir G and Zhang, Tianjun and Wang, Xin and Gonzalez, Joseph E. 2023. *Gorilla: Large language model connected with massive apis*. PID https://github.com/ShishirPatil/gorilla.

Qin, Yujia and Liang, Shihao and Ye, Yining and Zhu, Kunlun and Yan, Lan and Lu, Yaxi and Lin, Yankai and Cong, Xin and Tang, Xiangru and Qian, Bill and others. 2023. *Toolllm: Facilitating large language models to master 16000+ real-world apis*. PID https://github.com/OpenBMB/ToolBench.

Rastogi, Abhinav and Zang, Xiaoxue and Sunkara, Srinivas and Gupta, Raghav and Khaitan, Pranav. 2020. *Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset*. PID https://github.com/google-research-datasets/dstc8-schema-guided-dialogue.

Tang, Qiaoyu and Deng, Ziliang and Lin, Hongyu and Han, Xianpei and Liang, Qiao and Sun, Le. 2023. *ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases*. PID https://github.com/tangqiaoyu/ToolAlpaca.

Wang, Xingyao and Wang, Zihan and Liu, Jiateng and Chen, Yangyi and Yuan, Lifan and Peng, Hao and Ji, Heng. 2023. *MINT: Evaluating LLMs in Multi-turn Interaction with Tools and Language Feedback*. PID https://github.com/xingyaoww/mint-bench.

Xu, Qiantong and Hong, Fenglu and Li, Bo and Hu, Changran and Chen, Zhengyu and Zhang, Jian. 2023. *On the Tool Manipulation Capability of Open-source Large Language Models*. PID https://github.com/sambanova/toolbench.

Yang, Rui and Song, Lin and Li, Yanwei and Zhao, Sijie and Ge, Yixiao and Li, Xiu and Shan, Ying. 2023. *Gpt4tools: Teaching large language model to use tools via self-instruction*. PID https://github.com/AILab-CVC/GPT4Tools.

Zang, Xiaoxue and Rastogi, Abhinav and Sunkara, Srinivas and Gupta, Raghav and Zhang, Jianguo and Chen, Jindong. 2020. *MultiWOZ 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines*. PID https://github.com/budzianowski/multiwoz.