DRUGPILOT: LLM-BASED PARAMETERIZED REASONING AGENT FOR DRUG DISCOVERY

Kun Li[a,1], Zhennan Wu[a,1], Shoupeng Wang[b,1], and Wenbin Hu[a,*]

[a]School of Computer Science, Wuhan University
[b]School of Mathematics and Statistics, Wuhan University
{likun98, wuzhennan, wangshoupeng, hwb}@whu.edu.cn

ABSTRACT

In the field of AI4Science, large-scale language models (LLMs) show great potential to parse complex scientific semantics, integrate cross-disciplinary knowledge, and assist critical task research. However, in the field of drug discovery, despite the optimization through professional data pre-training, context window expansion, and internet search, the existing LLMs are still facing challenges such as massive multi-modal and heterogeneous data processing, domain knowledge dynamic updating delay, and insufficient confidence in predicting the results of complex computational tasks. To address these challenges, we propose the DrugPilot, an LLM-based agent with parameterized reasoning for drug discovery. DrugPilot addresses key limitations of traditional end-to-end LLM prediction approaches through its parametric inference architecture. This agent system supports major phases of the drug discovery pipeline, facilitating automated planning and execution of multi-stage research tasks. To address the critical challenge of multi-modal drug data analysis (incorporating both public datasets and user-submitted data), we developed an interactive parameterized memory pool. This innovative component standardizes real-world drug data into parametric representations, simultaneously enabling efficient knowledge retrieval in multi-turn dialogue while mitigating the information loss inherent in text-based data transmission. Additionally, we created a drug instruct dataset across 8 essential drug discovery tasks for model fine-tuning and evaluation. Based on the Berkeley function calling evaluation framework, DrugPilot demonstrated the most advanced tool calling capabilities on our drug discovery tool instruction dataset, outperforming existing agents (e.g., ReAct, LoT). Specifically, it achieves task completion rates of 98.0%, 93.5%, and 64.0% on simple, multiple, and multi-turn tasks, respectively. The code is available at: `https://github.com/wzn99/DrugPilot`.

Keywords large language model · agent · tool calling · drug discovery · reasoning

1 Introduction

With the rapid development of deep learning technologies, AI-assisted drug discovery is blooming as a transformative approach, which significantly improves the efficiency and accuracy of key steps in these complex tasks [1, 2]. For instance, generative AI Chemistry42 [3] enables efficient molecular design, multi-domain drug screening tools MolProphet [4] accelerates compound evaluation, and end-to-end AI platform DrugFlow [5] streamlines early-stage workflows. Compared to conventional approaches, these computational tools demonstrate superior predictive performance and significantly accelerate drug discovery timelines.

But despite the significant revolution, there remains a considerable gap between the potential of AI technologies and their practical application in drug discovery[6, 7]. Drug discovery is a multi-stage and time-consuming process involving a series of complex tasks [8], such as drug generation and optimization [9], target affinity prediction [10], and molecular property prediction [11, 12]. On the one hand, researchers in pharmaceuticals, biology, and other related fields often lack the technical expertise to operate the state-of-the-art (SOTA) deep learning models [13, 14]. Specifically, the models have strict requirements for the input format, and thus researchers need to spend considerable time on preprocessing real-world data and converting inconsistent data across different platforms [15, 16]. The lack of

Figure 1: **a** Four application scenarios of DrugPilot: zero-code integration of AI models, scalable data acquisition, coordinated multi-task processing, and accurate execution of 8 essential drug discovery tasks. **b** Comparison of LLMs or agents for drug discovery with our DrugPilot in real-world usage scenarios

knowledge in programming, computer hardware, and system operations significantly increases the cost for researchers to effectively utilize SOTA models, and in some cases, prevents their use altogether [17, 18]. Moreover, most existing platforms function as isolated tools without automated task planning and execution capabilities [19], whereas drug discovery typically involves multiple stages and demands seamless integration across tasks [20].

Recent studies have made progress in automating certain drug discovery tasks using large language models (LLMs) [21, 22, 23]. For example, DrugAgent [24] employs two agents as instructor and planner to establish an end-to-end machine learning pipeline from data acquisition to model evaluation, implying LLMs' potential in automating drug discovery processes. Another study, also named DrugAgent [25], integrates knowledge graphs with literature mining to build an intelligent agent system for drug repositioning. By harnessing the reasoning capabilities of LLMs, these systems have demonstrated their effectiveness in handling multi-source heterogeneous data. These findings suggest that LLMs can not only facilitate cross-domain communication but have the potential to automate drug discovery through intelligent execution [26].

Despite fine-tuning with agent frameworks or domain-specific data, leveraging LLMs' reasoning capabilities for automated drug discovery and accurate prediction still faces significant challenges. Primarily, the natural language output format of LLMs inherently limits their ability to precisely represent specialized entities (e.g., drug compounds, cell lines, and targets) or reliably predict their interactions. This limitation often induces hallucination issues and multi-modal reasoning fragmentation [26, 21, 23], manifesting as failed cross-modal semantic alignment between molecular graphs (Graph), bioactivity data (numerical matrices), and literature evidence (natural language). Furthermore, current LLM-driven tools predominantly focus on isolated tasks without end-to-end workflow management [26], compelling researchers to manually switch tools and integrate intermediate data, which substantially compromises human-AI collaboration efficiency [21, 23].

To address these challenges, we propose DrugPilot, an LLM-based agent with parameterized reasoning for drug discovery. DrugPilot fully covers the whole process of drug research and development, and can automatically plan and execute multi-stage research tasks defined by user queries. To meet the critical needs for precise extraction and analysis of multi-modal drug data (including public datasets and user-uploaded data), we propose an interactive parameterized memory pool (PMP). As a highly flexible component, the PMP transforms real-world drug data into standardized parametric representations. This ensures efficient knowledge retrieval in multi-turn dialogue while resolving the information loss problems typical of text-based transmission. To address common reasoning errors that LLMs encounter when reading PMP and calling tools, as well as the tendency of LLMs to forget the initial task during long conversations,

we further propose the feedback and focus (Fe-Fo) mechanism. Meanwhile, we construct a tool-calling dataset for drug discovery (TCDD), containing 2,800 samples across 8 key drug discovery tasks for model fine-tuning. Evaluated referring to Berkeley Function-Calling Leaderboard [27], DrugPilot achieves superior performance over existing methods ReAct [28], CoT [29] and Lot [30]. Specifically, the task completion rates achieve 98.0%, 93.5%, and 64.0% on three evaluation categories, increasing by 13.2%, 66.1%, and 80.3% respectively, compared to the SOTA agent ReAct. Our contributions are as follows:

1. We propose DrugPilot, a parameterized reasoning LLM agent that automates multi-stage task planning and execution throughout the drug discovery pipeline, addressing the inefficiencies of traditional manual workflows.

2. A parameterized memory pool (PMP) is used for extracting textual information and processing large-scale multi-modal data. The PMP employs a unified parametric representation for multi-modal data, while supporting both lossless information preservation and real-time interactive updating capabilities.

3. The Fe-Fo mechanism is designed to address reasoning errors in LLMs. Fe-Fo targets errors that occur when LLMs read PMP and call tools, providing specific error feedback to help LLMs correct their mistakes, and restating the original question to help LLMs maintain focus.

4. We construct a high-quality instruction dataset, TCDD, containing 2,800 annotated samples across 8 drug discovery tasks to improve domain-specific instruction understanding and tool calling accuracy of LLMs.

5. Referring to the Berkeley Function-Calling Leaderboard, a custom evaluation framework is constructed to test the performance of LLMs and agents. And the task completion rates of DrugPilot achieve 98.0%, 93.5%, and 64.0% on three evaluation categories, increasing by 13.2%, 66.1%, and 80.3%, respectively, compared to the SOTA agent ReAct.

2 Related Works

2.1 LLM for Drug Discovery

Large language models (LLMs) are now being applied to multiple core stages of drug discovery, including molecular optimization [31, 32] and property prediction [33, 34], effectively reducing R&D costs and accelerating the development process. These methods can be technically categorized into: agent systems that call external tools, and LLMs that operate without external tools. The tool-calling agent systems integrate specialized tool chains by generating API commands. Representative examples include DrugAssist [31] and MolecularGPT [34], which focus on molecular optimization and property prediction, respectively, featuring interactive human-machine dialogue frameworks. Notably, two distinct studies, both named DrugAgent, proposed different innovations: one employs a dual-agent architecture to establish an end-to-end machine learning pipeline [24], while the other combines knowledge graphs with literature mining for drug repositioning [25]. Although these methods can handle complex tasks, they remain limited by task specificity and small-sample constraints. LLMs can also work independently, using their natural language processing strengths to represent medical data in text form, which offers unique benefits for drug discovery. For example, DrugLLM [33] proposed a group-based molecular representation approach that generates molecules meeting target properties with minimal samples. Tx-LLM [35] achieves basic performance across multiple tasks by integrating multi-modal medical data. These models eliminate dependence on traditional tool chains, showing the advantages in convenience and generalization, which provides new technological pathways for drug development.

Despite showing theoretical promise in drug discovery, LLMs face significant challenges in practical applications regarding multi-task automation coordination [36, 37] and high-precision prediction [38, 39, 34] requirements. Moreover, LLMs depend on in-context learning and prompt engineering, which struggle to satisfy precise numerical constraints or optimize for specific quantitative targets [32]. For instance, in molecular generation and property prediction tasks, their lack of physical grounding may yield molecules with suboptimal fitness or produce invalid SMILES strings that fail to decode into chemically valid structures [40]. Furthermore, the fragmented tools and incompatible cross-platform data formats further hinder end-to-end workflow efficiency [41]. While existing LLMs perform well in literature analysis, breakthroughs in core tasks still require assistance from external computational tools.

2.2 LLM Tool Calling

In recent years, the capabilities of AI agents to solve complex real-world tasks have been significantly enhanced by equipping LLMs with external tools [42, 43, 44]. This integration of t calling enables LLMs to access real-time information, perform precise calculations, and invoke third-party services, thereby unlocking a wide range of application potentials across various domains [45].

The ReAct significantly mitigates hallucination issues in knowledge-intensive tasks by alternately generating reasoning steps and tool invocation instructions [28]. LLMCompiler employs a parallel tool calling architecture to improve task execution efficiency [46]. ToolACE constructs a diverse tool library containing 26,507 APIs, enabling an 8-billion-parameter model to achieve performance comparable to GPT-4 in tool invocation tasks [47]. Additionally, several reasoning frameworks have been proposed to enhance tool calling capabilities of LLMs, including Chain-of-Thought (COT) for step-by-step reasoning [29], Graph-of-Thought (GoT) for optimizing multi-path thinking through graph structures [48], and Tree-of-Thought (ToT) for exploring optimal solutions via tree structures [49].

Nevertheless, current agent systems face significant challenges in specialized domains like drug discovery [25, 24]. Approximately 82% of existing systems only support basic data types (e.g., strings, integers) with parameter nesting levels limited to two [50], making them inadequate for handling complex data types commonly encountered in drug research, such as long strings, floating-point numbers, and files.

2.3 Memory Mechanism of LLM

A large body of recent research has focused on LLM-based agents, equipping LLMs with additional modules to enhance their ability to operate in real-world environments [51]. Among these newly added modules, the memory component is a critical part, playing a vital role in several aspects, including how agents accumulate experience [52], explore the environment [53], and abstract knowledge [54].

Existing memory modules mainly adopt a text-based memory form [55]. For example, LongChat [56] uses complete interactions, which stores all information of the agent-environment interaction history, concatenating multi-turn dialogues, operations, and feedback into a long text input; SCM [57] and MemGPT [58] use recent interactions, which retains only the most recent interactions according to the Principle of Locality [59]; MemoryBank [60] and RET-LLM [61] use retrieved interactions, which dynamically retrieves past interactions relevant to the current task by vectorizing and computing similarities to select the most relevant memory.

In tool-calling scenarios for drug discovery, current memory mechanisms face significant limitations. Such tasks often involve large-scale, multi-modal drug-related parameters [21, 23, 26] and demand high accuracy when calling tools. However, existing memory approaches typically mix these parameters with other information, embedding everything within unstructured text, and lack integration between memory modules and prompt engineering. In practice, they simply append all retrieved text into the prompt [60, 61]. This leads to high computational costs due to the quadratic complexity of the attention mechanism [62] and increases the risk of context overflow. Moreover, because of the lack of task structure awareness, LLMs cannot reliably extract the required information from retrieved memories, making it difficult for them to track and extract scattered parameter information across multiple dialogue turns, thus resulting in unpredictable outputs. While many memory systems focus on summarization or reflection to generate high-level knowledge [60], drug discovery tasks require the precise extraction and complete transmission of parameters, which summarization cannot address. Moreover, due to the lack of a clear structure in memory representation and the inclusion of a large amount of unnecessary information, current designs suffer from poor user interpretability and operability, limiting users' ability to control the task process.

3 Tool-Calling Dataset for Drug Discovery

LLMs have achieved strong performance in general tool calling tasks, but due to the lack of knowledge of relevant fields [63], complex data forms, and the need for batch processing of data [64], LLMs still struggle to correctly infer tool names and parameters of drug discovery tasks. To address these challenges, we developed a tool-calling dataset for drug discovery [1] (TCDD) to fine-tune and test LLMs. Specifically, TCDD is used to validate the three key capabilities of LLMs:

1. Tool selection: comprehending natural language queries of drug discovery tasks and selecting corresponding drug discovery tools;

2. Parameter extraction: identifying and extracting required parameters of unique data forms from the context.

3. Interaction with PMP: retrieving parameterized data in PMP and saving execution results.

The TCDD simulates real-world conversations between users and AI systems in the field of drug discovery, comprising various scenarios such as single-turn, multi-turn, error correction, and memory pool updates. It has a total of 2,800 instruction samples, 2500 for training and 300 for testing, with the ShareGPT format, which is originally designed for multi-turn dialogue. This format supports LLMs in interacting with external services and thus is particularly suitable

---

[1]`https://drive.google.com/file/d/1JthOkIAzuuaajZhgHO3e9TfM9KwBHmni/view?usp=sharing.`

Tool Name	Sample			Parameter	Output	Dataset
	single-turn	multi-turn	error-parameter			
`drug_property`	500	200	50	$\mathcal{L}_{\mathrm{drug}}, \mathcal{S}_{\mathrm{property}}$	reg/cls	●●●●●
`drug_cell_response`	200	220	50	$[(\mathcal{S}_{\mathrm{drug}}, \mathcal{S}_{\mathrm{cell}})]$	reg	●
`drug_target_affinity`	250	140	100	$[(\mathcal{S}_{\mathrm{drug}}, \mathcal{S}_{\mathrm{target}})]$	reg	■■
`drug_target_interaction`	250	90	100	$[(\mathcal{S}_{\mathrm{drug}}, \mathcal{S}_{\mathrm{target}})]$	cls	■■
`drug_drug_interaction`	50	140	50	$[(\mathcal{S}_{\mathrm{drug}}, \mathcal{S}_{\mathrm{drug}})]$	reg/cls	●●
`drug_generation`	100	140	50	$\mathcal{C}^*$	molecules	▲▲···
`drug_optimization`	50	140	100	$\mathcal{S}_{\mathrm{drug}}, \mathcal{C}^*$	molecules	▲▲···
`synthetic_path`	100	130	50	$\mathcal{L}_{\mathrm{drug}}$	pathway	●

Notations:
① Dataset: ● BACE, ● BBBP, ● ESOL, ● FreeSolv, ● LIPO, ● GDSCv2 [65], ■ DAVIS [66], ■ KIBA [67], ■ BindingDB [68], ● DrugBank [69], ● TWOSIDES [70], ▲ ZINC [71], ▲ QM9 [72, 73] ● USPTO [74].
② In column parameter, the capital letters denote the types of corresponding parameters, with $\mathcal{L}$ standing for list and $\mathcal{S}$ standing for string.
③ In this study, we do not distinguish between drug, molecule, compound, etc., and denote them as drug.
④ Tools `drug_generation` and `drug_optimization` integrate multiple models trained on different datasets, and they are suitable for different conditions. We represent their parameters with $\mathcal{C}^*$ uniformly.

Table 1: Basic information of drug discovery tools including names, input/output parameter structures, data sources, and the specific quantities of samples in the training dataset.

for drug discovery tasks characterized by multi-turn workflows and tool calling. Table 1 shows the basic information of the eight drug discovery tools in the TCDD, containing their names, input/output parameter structures, data sources, and the specific quantities of samples for different dialogue patterns in the training set.

Since the interaction with the memory pool requires formalized parameters, the samples are categorized into two types based on whether the memory pool mechanism is employed. Figure 2 illustrates a single-turn dialogue example using the `drug_property` tool to predict the aqueous solubility of a given drug. Each sample consists of three components: conversation, system instructions, and tools, with variations in the conversation portion between the two sample types. The conversation segment includes:

1. Human: The user inputs a natural language description of the drug discovery task;
2. Function Call: The LLM generates the tool calling information in JSON format, specifying the tool name and corresponding parameters;
3. Observation: The tool is called and the execution result is returned;
4. LLM: The LLM generates a response integrating the observation and task objectives.

The system section guides the LLM to execute tool calling and return results in a standardized format (see Figure 3 for details). The tools section provides essential information about the available tools, including: function names and descriptions, acceptable parameters with types and descriptions, and required parameters.

During the development of TCDD, we designed single-turn/multi-turn dialogue patterns, diversified user query expressions, and varied drug molecule types to enhance performance. Approximately 50% of the samples consist of single-turn dialogues with complete instructions, establishing fundamental tool calling rules and memory pool retrieval mechanisms. The remaining samples simulate complex multi-turn dialogues reflecting real-world workflows, including multi-turn tasks (30%) and parameter error scenarios (20%). For instance, in a typical drug optimization workflow, the system first generates molecular candidates based on inhibitory concentrations against specific cell types, then predicts drug properties (e.g., solubility), evaluates drug-target binding affinity, and finally refines the candidates. Such workflows require the model to integrate intermediate results from prior steps, efficiently extract parameters using the memory pool, and handle error propagation when tool failures occur. For common user input issues such as missing or misspelled parameters, the tool calling returns observations containing specific error messages. Corresponding samples in the dataset train the LLM to accurately discern user intent and interact appropriately based on tool specifications and observations.

The distribution of samples across different tools and patterns was determined considering their usage scenarios, parameter complexity, and frequency of application. In a complete workflow, tools such as `drug_optimization` and `drug_generation` are more likely to be used in combination with others. Hence, their ratios of multi-turn to single-turn samples are relatively higher. Since `drug_target_affinity` and `drug_target_interaction` require target protein sequences as parameters, which are typically long and complex, these tools have more error-correction samples. Similarly, given that `drug_optimization` involves significantly more parameters than other tools, it also includes a higher proportion of error-handling samples.

Human:
Predict the water solubility for the drug 'CC/C=C/C'.

Function Call:
```
{
    'name': 'drug_property_prediction',
    'arguments': {
        'drug_smiles': ['CC/C=C/C'],    → Actual value
        'property': 'logSolubility'
    }
}
```

Observation:
```
[{
    'smile': 'CC/C=C/C',
    'value': -0.11
}]
```

LLM:
The water solubility of drug 'CC/C=C/C' is -0.11 log(mol/L). Water solubility is an important factor in drug design, especially in the drug's bioavailability and efficacy.

(a) Conversations without memory pool

Human:
Predict the water solubility for the drug I loaded.

Function Call:
```
{
    'name': 'drug_property_prediction',
    'arguments': {
        'drug_smiles': '(user_smiles)',    → Key in PMP
        'property': 'logSolubility'
    }
}
```

Observation:
```
[{
    'smile': 'CC/C=C/C',
    'value': -0.11
}]
```

LLM:
The water solubility of drug 'CC/C=C/C' is -0.11 log(mol/L). Water solubility is an important factor in drug design, especially in the drug's bioavailability and efficacy.

(b) Conversations with memory pool

System:

You are a function calling AI model. You are provided with......

For each function call return a json object with......

Tools:
```
[{
    'name': 'drug_property_prediction',
    'description': 'Predict properties of drugs based on SMILES',
    'parameters': {
        'type': 'object',
        'properties': {......}
    },
    'required': [ 'drug_smiles', 'property' ]
}]
```

(c) The system and tools fields in the conversations.

Notations:
① There are different formalized parameters corresponding to keys in the memory pool, this sample takes *(user_smiles)* for an example.
② *Function Call* is the proper key in ShareGPT format, and we identify function calling and tool calling in this paper.

Figure 2: Sample structure with ShareGPT format for simple-turn dialogue. The dataset samples following the ShareGPT format comprise conversations, system, and tools. a) Conversations part without memory pool: These four elements may appear multiple times in multi-turn dialogue. b) Conversations part with memory pool: In tool calling, LLM uses formal parameter *(user_smiles)* to extract parameters from the memory pool. c) System part and tools part: In the system, there is a system prompt about the role of LLM and the output format. In the tools, there is basic information about the tool, especially the required parameters.

System Prompt

```
# Role definition
You are a function calling AI model. You are provided with function signatures within
<tools></tools> XML tags. You may call one or more functions to assist with the user query.
Don't make assumptions about what values to plug into functions.
# Output format requirements
For each function call return a json object with function name and arguments within
<tool_call></tool_call> XML tags as follows: <tool_call>{\"name\": <function-
name>,\"arguments\": <args-dict>}</tool_call>
```

Figure 3: System prompt. This prompt is used to guide LLMs to call tools in the correct format.
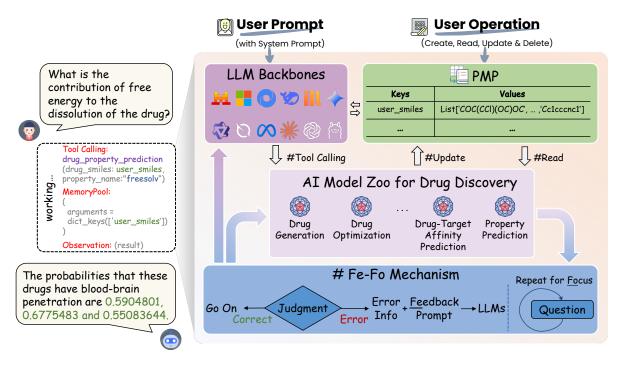
Figure 4: The framework of DrugPilot. DrugPilot comprises four components: the LLM backbones, the PMP, the AI models tailored to 8 stages of drug discovery with tool calling support, and the Fe-Fo mechanism.

4 Methods

4.1 Overview

To optimize the performance of LLM Agents in executing drug discovery tasks through tool-calling, we design DrugPilot. First, we constructed the TCDD dataset and used it to fine-tune LLMs, aiming to enhance the models' domain-specific knowledge in drug discovery and their understanding of drug-related tools. Second, we propose the parameterized memory pool mechanism to handle large-scale, multi-modal data and multi-turn conversations. Finally, we propose a feedback and focus mechanism to help LLMs correct common errors when calling drug-related tools and to maintain focus on the original task.

The DrugPilot framework, depicted in Figure 4. The user's drug discovery task is first provided as input to the LLM backbones. These LLMs will have the reason to make decisions on reading the PMP and calling tools. After verification by the Fe-Fo mechanism, the tool calls are executed, and results are returned. If errors are detected during verification, feedback is provided to the LLMs through the Fe-Fo mechanism. Throughout this process, users can view or update parameters in the PMP at any time to oversee the task execution. The LLMs will continuously reason based on the above information until the final answer is produced.

4.2 LoRA Fine-Tuning

To enhance the comprehension of LLMs regarding the tools integrated within DrugPilot and their respective parameters, prompt engineering techniques were employed. This involved augmenting the system prompt with detailed descriptions for each tool. Furthermore, to enhance the LLM's ability to call tools, its knowledge level in drug discovery, its understanding of our custom tools, and its ability to utilize the memory pool component, we constructed the TCDD dataset and performed LoRA fine-tuning on a series of 7B–9B LLM backbones. Based on evaluation feedback, the dataset has been iteratively optimized through multiple versions.

Post-processing After Fine-tuning   After fine-tuning, the LLMs showed significantly improved understanding of drug discovery and generated more usable outputs, but exhibited decreased format compliance. Common formatting errors included: extra whitespace in outputs, duplicate "Answer" fields, incorrect content delimiters, and so on. To extract usable content from these non-compliant outputs, we designed a regular-expression-based output parser. By implementing more tolerant parsing rules, this post-processing step effectively addressed the LLMs' common formatting
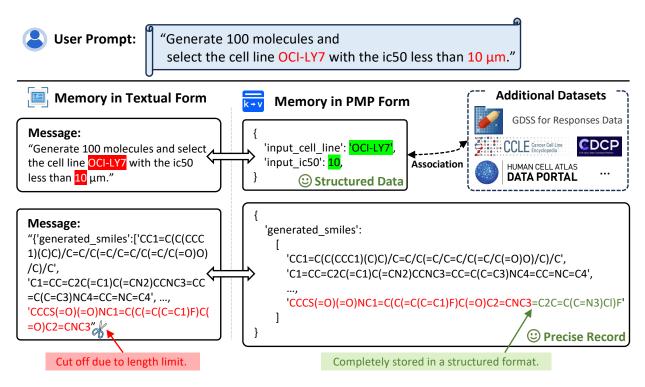
Figure 5: PMP's structure. In traditional memory modules, memory takes the form of a series of conversation text fragments, with drug-related parameters embedded within the text. In practical use, the scale of these molecular expressions will far exceed the examples, imposing a significant burden on memory retrieval and LLM reasoning. In our PMP, memory is stored as key-value pairs, where LLMs interact only with concise keys, while tools directly interact with the structured values.

errors. This approach reduced the strictness of format compliance requirements while significantly improving task completion accuracy.

4.3 Parameterized Memory Pool

To address the limitations of traditional memory modules in transmitting large-scale, multi-modal drug-related parameters, we propose a novel memory module: the parameterized memory pool (PMP). PMP does not store unstructured text but instead maintains structured key-value pairs. Its purpose is to optimize parameter passing efficiency within the agent, thereby reducing the reasoning burden on LLMs, improving the controllability of reasoning results, and ultimately enhancing the performance of LLMs in drug-related tool calling. We conducted prompt engineering to help LLMs better understand PMP. The memory pool prompt explicitly defines the purpose, usage scenarios, and usage method of PMP, as detailed in Appendix A.

The Structure of PMP   PMP is a key-value store, where each key-value pair maintains a parameter related to drug discovery. Each key is unique and is a short string representing the name of a drug-related parameter, designed to convey its meaning as clearly as possible for interaction with LLMs. The value is the actual content of the parameter, structured in a format that can be directly used as input for drug-related tools, enabling seamless interaction between PMP and external tools. When there are multiple parameters of the same type, they share one key, and their corresponding values are stored as a list containing all the parameters of that type.

As shown in Figure 5, when storing a large-scale list of drug molecules, the key could be `"input_drug_smiles"`, indicating that it represents a user-provided list of molecular expressions. Such drug-related parameters can be of considerable scale, for example, they may consist of lists with tens of thousands of entries and be embedded within complex textual contexts. This large-scale, multi-modal context poses significant challenges for both storage and reasoning. The PMP extracts these large-scale parameters from complex text and converts them into concise keys, allowing LLMs to interact only with these short keys. This greatly reduces the length of context required for the task.

Furthermore, PMP maintains the parameters in a structured manner, enabling subsequent tool invocations to directly use these parameters without requiring LLMs to repeatedly infer their complete content.

Parameter Reading and Updating Mechanism In traditional memory modules, the stored memory content is typically text fragments from the conversation. When performing reasoning with the LLMs, the user inputs a piece of text, memory modules will also retrieve a piece of text from the stored historical information. These two parts of the text are concatenated and fed into the LLMs together for reasoning. This parameter extraction process can be formulated as:

$$v = \mathcal{R}_{LLM}(T_u + T_m), \quad \text{where } T_m \subseteq \mathcal{M} \tag{1}$$

where, $\mathcal{M}$ denotes the entire stored memory, T_m denotes the retrieved memory text, T_u denotes the user input text, $\mathcal{R}_{LLM}$ denotes the function for retrieving parameters from text using LLMs and v denotes the final parameter passed to the tool. In this method, the large-scale complex texts bring a tremendous inference burden for LLMs

Figure 6 illustrates the workflow of the PMP. First, before the conversation begins, users can upload their own datasets or public datasets into DrugPilot. It is important to note that users are not limited to uploading parameters only at the beginning. They can add, delete, modify, and query parameters in the PMP at any time. This ability to dynamically control and adjust the task direction at any stage enhances the flexibility of the memory module. Compared to disorganized large blocks of text, the structured memory format enables users to operate on memory content. This feature allows humans to efficiently interact with the memory of large models and significantly improves the flexibility of the memory module.

Next, after the drug discovery task and the current keys stored in PMP are input into the LLM, the LLM analyzes the input text, selects the tool to call, and considers how to obtain the parameters required by that tool. If the required parameters are included in the input text, the LLM will extract them from the text. At this point, PMP saves the parameter as a key-value pair by assigning a key to the parameter and storing its content in the corresponding value for use in the next step. If the key already exists, PMP will use a list as the value for that key and append the current parameter to the list.

If the user's input text does not contain the required parameter, the LLM will select a key corresponding to the required parameter from the list of keys in the PMP. The PMP will then map the selected key to its corresponding value and pass that value as a parameter to the tool. If the value is a list containing multiple parameters, the PMP will take the last element of the list, which is the most recently added parameter. Through the key-value conversion mechanism of PMP, LLMs' reasoning task becomes selecting the key to use. It makes LLMs no longer interact directly with large-scale data, effectively reducing the burden on various components of the LLM-based agent system, including LLM inference, memory storage, and memory retrieval. The parameter extraction process of DruPilot can be formulated as:

$$v = \mathcal{G}(\underset{k \in \mathcal{M}_\mathcal{K}}{\arg\max} \, \mathcal{P}_{LLM}(\mathcal{M}_\mathcal{K})) \tag{2}$$

where, $\mathcal{M}_\mathcal{K}$ denotes the key set in the memory pool, k denotes a key from $\mathcal{M}_\mathcal{K}$, $\mathcal{P}_{LLM}$ denotes the function using LLMs to get the probability of using k, and $\mathcal{G}$ denotes the predefined mapping function. The PMP provides the LLMs with a series of selectable parameter keys.

After a successful tool calling, PMP will save the result returned by the tool as a key-value pair. It assigns a key to the result and stores the entire result as the value. In the next step, this result will be added to PMP's key list and passed as input to the LLM. At this point, the LLM can choose this execution result as the input parameter for the next tool invocation.

4.4 Fe-Fo Mechanism

LLMs cannot always perfectly select tools and pass parameters with complete correctness in both content and format. Additionally, LLMs tend to forget the initial task during long conversations. To address these issues, we propose a feedback and focus mechanism, named Fe-Fo, to help LLMs correct errors and maintain focus, making their output more controllable.

In the feedback mechanism, we alleviate common reasoning issues by feeding error information back to LLMs. When calling drug-related tools, the reasoning output of LLMs exhibits a series of common issues, as illustrated in Appendix B. We have verified these issues and designed corresponding feedback prompts for each error type. These prompts describe the type and cause of the error in detail and explicitly instruct the LLMs to regenerate the output according to the requirements based on this information.

In the focus mechanism, we help LLMs stay focused by reiterating the original task. If LLMs are unable to resolve the issue in one attempt, the length of the conversation will inevitably increase. At this point, LLMs often forget the initial
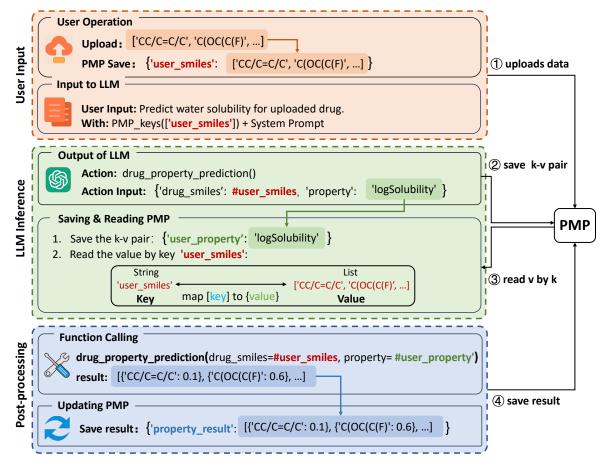
Figure 6: Sample Workflow of PMP during a conversation. ① The user uploads a series of drug molecules. Users can view or update the PMP at any time, allowing them to control the progress of the task. ② The PMP automatically stores the parameters extracted by the LLMs. ③ The PMP maps the key selected by the LLMs to their corresponding values and passes them to the tool. ④ The PMP saves the returned parameters from the tool after the tool is called.

task, leading to a loss of focus and resulting in aimless attempts, such as randomly changing the selected tool or passing parameters based on hallucinations. To address this issue, when LLMs make a reasoning error, DrugPilot will repeat the original task to the LLMs, ensuring that they remain focused on the required drug discovery task.

Fe-Fo integrates the above two types of information and provides a clear instruction: regenerate the output according to the formatting requirements based on the given information. When any error from the types shown in Appendix B is detected, the Fe-Fo mechanism will feed the Fe-Fo prompt into the LLMs, which can be formulated as:

$$O_{t+1} = LLM(\mathcal{E}(O_t) + T + I) \tag{3}$$

where, I denotes the instruction to guide the reasoning, T denotes the original task description, $\mathcal{E}$ denotes the error detection function applied to the output of LLMs which returns the feedback information corresponding to the error and O_t denotes the output of LLMs at step t.

5 Experiments

5.1 Settings

We fine-tuned a series of small-scale LLMs on TCDD using LoRA, including Meta-Llama-3.1-8B-Instruct [75], Meta-Llama-3-8B-Instruct [76], Mistral-Nemo-Instruct-2407 [77], Gemma-2-9B-it [78], Qwen2-7B-Instruct [79], DeepSeek-LLM-7B-Chat [80], DeepSeek-R1-Distill-Llama-8B [81] and Llama-3-Groq-8B-Tool-Use[2]. We use the

---

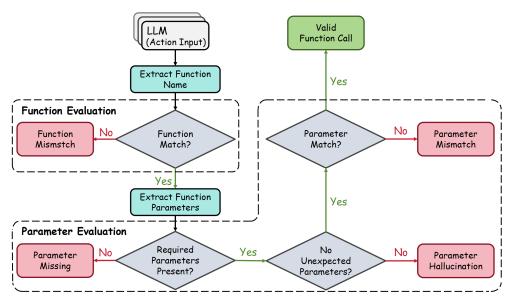[2]https://groq.com/introducing-llama-3-groq-tool-use-models/.

Figure 7: The step-by-step evaluation process. The action inputs from LLM go through function evaluation and parameter evaluation sequencely, and are compared with the true labels. The two parts of the evaluation correspond to function accuracy and parameter accuracy in the final results.

open-source dataset glaive_toolcall[3] to improve the general tool-calling capabilities of the LLMs. And we divided our tool-calling dataset for drug discovery (TCDD) into training, validation, and test sets at an 8:1:1 ratio to enhance LLMs' specialized capabilities in drug-related tool calling and evaluated their performance. The hyperparameter settings used during the fine-tuning process are detailed in the Appendix C.

Baselines DrugPilot integrates multiple drug discovery tools, and it mainly focuses on accurate tool calling to complete a full workflow of drug discovery tasks. However, some existing methods, like DrugAgent [25], are dedicated to improving performance on individual tasks, concentrating on text understanding and result interpretation. These methods do not align well with our task objectives. Therefore, we select three representative agents with tool-calling capabilities in recent times for comparison:

- CoT [29] is a simple baseline where the agent generates a solution by breaking the problem into substeps.
- LoT [30] is a self-improvement prompting framework where the agent verifies and refines its intermediate reasoning steps by embedding symbolic logic principles.
- ReAct [28] interleaves reasoning traces with actions to enable LLMs to iteratively plan, gather information, and adjust strategies.

And these three baseline agents are based on pre-trained LLMs.

Evaluation metrics Drawing on the Berkeley Function-Calling Leaderboard [27], we designed a custom evaluation framework for DrugPilot and other agents. A valid tool calling requires correct tool selection, accurate parameter extraction, and effective self-correction across multiple turns. Therefore, we divide the evaluation into three categories, with 100 queries in each category, which together constitute the test set.

- Simple function: This category contains the simplest situation with one and only one function supplied.
- Multiple function: This category contains a user query that requires the invocation of only one function among eight available tools.
- Multi-turn function: This category contains multi-turn queries, where different queries may correspond to different tool callings.

For each task, the LLM generates an action input, information required to call the tools, in JSON format. Given that hallucination remains a challenge for LLMs, especially in tool calling, we evaluate their responses in two aspects: tool

---

[3]https://huggingface.co/datasets/glaiveai/glaive-function-calling-v2.

11

Category	Model	Scale	DrugPilot (ours)		DrugPilot w.o. SFT		CoT		LoT		ReAct	
			Acc.F	Acc.P	Acc.F	Acc.P	Acc.F	Acc.P	Acc.F	Acc.P	Acc.F	Acc.P
Simple Function	Llama3.1	8B	**98.4**$_{1.2}$	**98.0**$_{1.3}$	89.4$_{0.4}$	88.4$_{1.0}$	92.6$_{2.9}$	63.4$_{3.4}$	81.2$_{5.1}$	46.4$_{4.6}$	86.6$_{4.4}$	86.6$_{4.4}$
	Llama3	8B	**99.2**$_{0.8}$	**97.6**$_{2.1}$	73.0$_{3.9}$	73.0$_{3.9}$	81.0$_{3.5}$	75.0$_{3.2}$	43.4$_{4.5}$	38.8$_{5.3}$	71.0$_{4.2}$	71.0$_{4.2}$
	Mistral-NeMo	7B	**97.4**$_{0.5}$	**97.0**$_{1.0}$	89.2$_{1.8}$	89.2$_{1.8}$	90.8$_{4.9}$	89.0$_{5.4}$	93.4$_{2.8}$	89.4$_{4.2}$	88.6$_{4.8}$	88.6$_{4.8}$
	Gemma2	9B	**97.8**$_{1.1}$	**97.4**$_{1.1}$	90.2$_{0.1}$	90.2$_{0.1}$	87.0$_{0.1}$	86.4$_{1.1}$	85.8$_{1.8}$	80.6$_{1.1}$	88.0$_{0.1}$	87.6$_{0.5}$
	Qwen2	7B	**99.3**$_{0.1}$	**95.9**$_{1.8}$	98.6$_{0.1}$	89.2$_{3.9}$	98.2$_{1.1}$	87.8$_{1.5}$	97.0$_{1.6}$	83.2$_{3.1}$	97.4$_{0.5}$	88.4$_{2.3}$
	DeepSeek-LLM	7B	**47.6**$_{2.6}$	**41.2**$_{4.0}$	33.9$_{4.7}$	33.6$_{4.1}$	17.2$_{4.7}$	13.6$_{2.7}$	19.6$_{3.6}$	11.2$_{4.0}$	40.4$_{3.4}$	27.4$_{3.6}$
	DeepSeek-R1*	8B	**97.2**$_{0.8}$	**73.6**$_{6.2}$	66.2$_{0.1}$	61.8$_{1.3}$	48.2$_{5.4}$	43.6$_{4.2}$	61.0$_{3.3}$	54.8$_{2.6}$	58.0$_{4.6}$	56.6$_{4.3}$
	Llama3-Groq	8B	**99.8**$_{0.4}$	**90.6**$_{2.4}$	44.0$_{5.6}$	43.0$_{5.4}$	15.8$_{1.9}$	15.2$_{2.3}$	23.2$_{2.9}$	22.6$_{3.2}$	38.6$_{4.5}$	38.6$_{4.6}$
Multiple Function	Llama3.1	8B	**98.7**$_{0.6}$	**93.5**$_{2.3}$	79.4$_{2.9}$	58.2$_{1.2}$	55.8$_{3.3}$	50.5$_{3.4}$	53.5$_{2.7}$	43.0$_{3.6}$	76.3$_{5.1}$	56.3$_{5.6}$
	Llama3	8B	**98.0**$_{2.3}$	**92.8**$_{4.6}$	65.0$_{1.7}$	54.3$_{2.0}$	46.3$_{5.4}$	43.3$_{5.6}$	35.8$_{4.6}$	32.8$_{4.2}$	59.8$_{2.1}$	53.5$_{2.7}$
	Mistral-NeMo	7B	**98.4**$_{2.1}$	**96.3**$_{2.5}$	86.8$_{4.8}$	83.7$_{3.8}$	82.0$_{2.7}$	68.3$_{3.4}$	76.0$_{3.8}$	70.8$_{3.0}$	92.3$_{3.5}$	77.8$_{3.4}$
	Gemma2	9B	**99.5**$_{0.7}$	**92.2**$_{2.4}$	91.0$_{2.2}$	72.4$_{3.1}$	92.5$_{2.7}$	79.8$_{1.9}$	86.8$_{2.9}$	76.3$_{4.2}$	93.0$_{1.4}$	73.8$_{2.3}$
	Qwen2	7B	**93.7**$_{2.9}$	**87.2**$_{2.3}$	93.2$_{2.9}$	76.0$_{5.7}$	87.5$_{2.5}$	77.8$_{4.5}$	83.3$_{4.6}$	71.5$_{3.0}$	92.5$_{1.1}$	74.5$_{1.9}$
	DeepSeek-LLM	7B	**39.5**$_{2.7}$	**18.8**$_{1.6}$	26.5$_{6.5}$	13.0$_{2.1}$	11.0$_{2.1}$	5.3$_{2.1}$	9.5$_{3.3}$	5.0$_{3.5}$	20.5$_{2.4}$	8.5$_{2.2}$
	DeepSeek-R1	8B	**93.9**$_{2.0}$	**72.2**$_{4.1}$	60.3$_{5.3}$	49.0$_{2.2}$	51.8$_{3.4}$	34.3$_{3.3}$	39.0$_{3.6}$	29.0$_{3.8}$	60.3$_{4.8}$	43.3$_{4.0}$
	Llama3-Groq	8B	**96.2**$_{0.9}$	**78.5**$_{2.1}$	39.5$_{4.2}$	30.5$_{3.2}$	17.5$_{3.5}$	17.0$_{3.1}$	10.3$_{1.9}$	9.3$_{1.4}$	40.3$_{7.4}$	28.8$_{4.6}$
Multi-turn Function	Llama3.1	8B	**72.7**$_{6.7}$	**64.0**$_{5.7}$	49.5$_{2.4}$	38.2$_{2.3}$	43.0$_{3.6}$	31.8$_{2.9}$	30.1$_{5.4}$	24.5$_{4.8}$	45.1$_{5.2}$	35.5$_{3.7}$
	Llama3	8B	**74.2**$_{4.1}$	**61.9**$_{4.7}$	34.8$_{4.7}$	25.1$_{5.4}$	25.8$_{5.9}$	17.3$_{3.8}$	20.7$_{3.3}$	14.7$_{2.6}$	32.6$_{1.4}$	18.0$_{2.6}$
	Mistral-NeMo	7B	**70.0**$_{6.1}$	**56.9**$_{4.1}$	54.9$_{5.1}$	37.2$_{2.9}$	44.4$_{3.2}$	33.2$_{3.1}$	38.9$_{3.8}$	31.3$_{1.6}$	50.9$_{2.9}$	39.5$_{3.0}$
	Gemma2	9B	**84.9**$_{1.9}$	**61.8**$_{4.1}$	44.3$_{1.8}$	34.4$_{2.1}$	54.3$_{2.3}$	43.1$_{2.6}$	35.3$_{3.4}$	27.6$_{3.1}$	34.5$_{2.1}$	32.8$_{1.9}$
	Qwen2	7B	**73.9**$_{3.2}$	**57.3**$_{3.3}$	38.2$_{4.0}$	22.7$_{2.6}$	34.2$_{3.9}$	20.1$_{2.6}$	35.2$_{3.8}$	20.2$_{3.9}$	21.4$_{1.1}$	18.7$_{1.2}$
	DeepSeek-LLM	7B	**19.9**$_{4.5}$	**9.0**$_{2.3}$	4.1$_{1.0}$	2.4$_{0.3}$	1.8$_{0.8}$	0.4$_{0.5}$	1.7$_{0.5}$	0.5$_{0.7}$	9.7$_{2.0}$	1.9$_{0.3}$
	DeepSeek-R1	8B	**71.5**$_{1.8}$	**51.4**$_{3.8}$	25.6$_{2.3}$	19.1$_{2.5}$	27.4$_{4.2}$	17.9$_{2.2}$	7.4$_{1.9}$	6.8$_{2.3}$	16.3$_{3.9}$	15.2$_{4.3}$
	Llama3-Groq	8B	**79.6**$_{3.4}$	**49.0**$_{5.1}$	24.4$_{1.9}$	19.4$_{2.2}$	8.9$_{1.2}$	7.1$_{1.0}$	8.5$_{2.5}$	5.6$_{1.5}$	12.4$_{3.0}$	11.8$_{3.4}$

Notations:
① The model Deepseek-R1 with 8B parameters stands for DeepSeek-R1-Distill-Llama-8B.
② The two accuracy rates, Acc.F and Acc.P are the average results of repeat evaluation experiments with the standard deviation attached as a corner mark.
③ The bolded values represent the highest accuracy rate, and the underlined values represent the second-highest accuracy rate.
④ In the multi-turn category, more weight is put on the later queries and thus its accuracy, or score, is more suitable, reflecting more authentic performance on multi-stage tasks.

Table 2: Overall results of experiments. The tool calling performance on three categories of different LLMs and agents are measured by two accuracy metrics. CoT, LoT, and ReAct are baseline methods, and DrugPilot is combined with SFT and pre-trained LLMs.

selection and parameter extraction. According to this, we report two accuracy metrics: function accuracy (**Acc.F**) and parameter accuracy (**Acc.P**). Additionally, detailed calculation formulas are shown in Appendix D.

Figure 7 shows the step-by-step evaluation process of the LLM response, namely the action input. First, the action input undergoes function evaluation, where the tool name is extracted and verified. Once the tool name is correct, the function parameters are parsed and undergo parameter evaluation. This step verifies whether all required parameters are present and that no unexpected parameter is included. Finally, each parameter value is inspected to confirm its correctness and integrity. Only when all these checks are successfully completed is the tool calling deemed valid. In addition, the maximum allowable execution time for an individual query is set to 120s; any response that exceeds this threshold is directly determined erroneous.

5.2 Overall Experiments

To fully understand the performance of DrugPilot, we conduct evaluation on different LLMs and different agents, and the results are reported in Table 2. DrugPilot achieves the highest accuracy on the three categories over all agents. For the simple function category, mainstream LLMs such as Llama3.1, Llama3, Mistral-NeMo, Gemma2, and Qwen2 all exceed 97% on Acc.F and 95% on Acc.P. In the multi-function category, DrugPilot continues to dominate. Across Llama3.1, Llama3, Mistral-NeMo, and Gemma2, Acc.F stays above 98% and Acc.P stays above 92%. Even in the most complex category, multi-turn function, DrugPilot achieves superior performance, attaining accuracy rates exceeding 70% and 60% compared to the SOTA methods.

As the difficulty of tool calling scenarios increases, the accuracy declines overall. But on all three categories, DrugPilot outperforms the SOTA agent ReAct by 13.6% and 13.2% in simple function, 29.4% and 66.1% in multi-function, 61.2% and 80.3% in multi-turn function on Acc.F and Acc.P. Notably, though less capable LLM like deepseek-llm-7b-chat performs poor, especially in the category of multi-turn function, it still gains improvement compared to baseline methods. Besides, after removing SFT, the performance of DrugPilot declines obviously, but it is still superior to the baseline methods in most cases. Compared with the second-best method, DrugPilot still has considerable improvement, with 6.3% and 10.9% in simple function, 24.3% and 60.6% in multi-function, 46.9% and 67.5% in multi-turn function on Acc.F and Acc.P.

In addition to measuring accuracy, we also recorded the average execution time for each query in the multi-turn function category. As shown in Figure 8c, the average latency of DrugPilot has been reduced to under 20s, and for models including Qwen2, DeepSeek-R1, Llama3, and Llama3.1, this means more than a twofold improvement over baseline
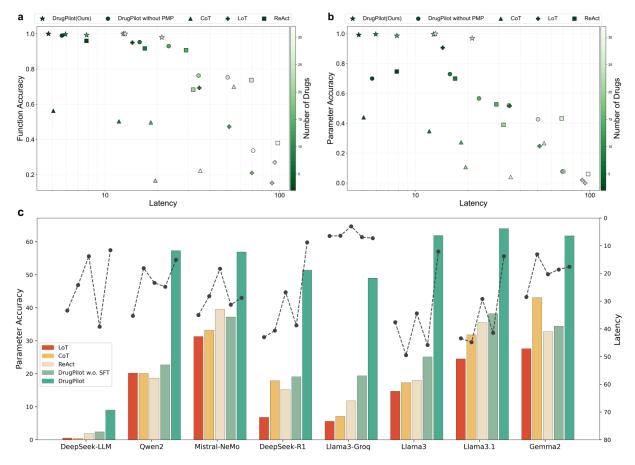
Figure 8: Visualization of the experiments. We tested how the number of drugs, ranging from 2 to 20 with an interval of 2, affects LLM performance. Function and parameter accuracy with latency of Llama3.1-8B are shown, with horizontal dashed lines representing results with PMP. **a** Function accuracy of different agent methods under varying molecule quantities. **b** Parameter accuracy of different agent methods under varying molecule quantities. **c** The parameter accuracy and latency of LLMs and agents on category multi-turn function.

methods. The decrease is particularly striking given that the multi-turn tasks require sequential reasoning based on former results. In practice, this means that DrugPilot not only delivers substantially higher parameter accuracy but does so with a runtime that is less than half of what SOTA agents require. The combination of SFT and an optimized tool-calling procedure allows DrugPilot to maintain fast response times even as task complexity grows, demonstrating that accuracy and efficiency gains can be achieved simultaneously.

5.3 Parameter Scale Experiments

The excessive scale of parameters is a key challenge in utilizing LLM-based agents for drug discovery tasks. To evaluate the upper limit of the parameter scale that existing methods can handle when processing drug-related parameters, we conducted a parameter scale experiment. We tested ChatGPT-4o[82], one of the most powerful closed-source large-scale LLMs, in a drug tool-calling scenario. We compared its performance with DrugPilot by invoking ChatGPT-4o's tool-calling API. The evaluation task involved calling a drug property prediction tool to predict the water solubility of molecules. By adjusting the number of drug molecules and their average string length in the input, we investigated its capability to handle large-scale drug parameters.

The experimental results are shown in Table 3. When the number of input molecules was ≤ 51 and the average molecular length was ≤ 90, ChatGPT-4o could stably select the correct tool, accurately pass the parameters, and ensure successful task execution. However, when these thresholds were exceeded, constrained by the model's context length, ChatGPT-4o failed to extract and output such large-scale parameters. These results indicate that ChatGPT-4o's

13

SMILES Count		1	20	50	51	51	52	52
Avg. SMILES Length		90	90	90	90	91	90	91
Task Completion	ChatGPT-4o	✓	✓	✓	✓	✗	✗	✗
	DrugPilot	✓	✓	✓	✓	✓	✓	✓

Table 3: Result of parameter scale experiments. 'SMILES Count' refers to the number of input drug molecules, namely the length of the molecule list. 'Avg. SMILES Length' refers to the average string length of the input drug SMILES. Task Completion indicates whether the task is completed successfully, including both selecting the correct tool and passing the corresponding parameters accurately.

SFT	Fe-Fo	PMP	Metric		
			Acc.F	Acc.P	Latency
✗	✗	✗	66.3	48.8	30.91
✗	✗	✓	85.0	50.4	27.04
✗	✓	✗	82.1	70.0	33.40
✗	✓	✓	92.9	59.2	25.74
✓	✗	✗	73.8	67.5	14.99
✓	✗	✓	91.3	88.3	11.61
✓	✓	✗	95.0	93.7	15.18
✓	✓	✓	98.7	96.1	9.29

Table 4: Ablation experiment: The two accuracy rates and the latency on category multiple function in the absence of SFT, Fe-Fo, and PMP based on Llama3.1-8B are displayed in this table.

performance in drug property prediction tasks is limited by the scale of input drug molecules, which must be considered in practical applications involving large-scale parameters in drug discovery.

In contrast, DrugPilot has no upper limit on parameter scale. Even with 91 molecules and an average length of 52, it could still accurately pass the parameters to the tool. Since PMP (Parameter Management Protocol) employs a key-value pair conversion mechanism to remove large-scale parameters from the LLM's context, DrugPilot can theoretically handle parameters of any scale. This effectively solves the critical issue of large-scale parameter transfer in drug discovery agents.

5.4 Ablation Study

To comprehensively evaluate the impact of each component within DrugPilot on overall system performance, we conducted ablation studies from three perspectives: Supervised Fine-Tuning (SFT), the Feedback-Forecast (Fe-Fo) mechanism, and the PMP (Parameterized Memory Pool). These experiments aim to analyze the individual contribution and interplay of each module. All experiments were performed using Llama3.1-8B as the foundation LLM under a simplified tool-calling scenario.

Effect of SFT and Fe-Fo Mechanism: To investigate the role of SFT and Fe-Fo, we removed each module separately while keeping all other conditions unchanged. We then evaluated the accuracy of tool selection and parameter extraction across multi-functional tasks. As shown in Table 4, both modules have a significant positive effect on performance. With the addition of SFT and Fe-Fo, tool-selection accuracy achieves 95.0% and parameter-extraction accuracy achieves 93.7%, increasing by 28.7% and 44.9% respectively. Notably, the latency has also been greatly improved, from the original 30.91s to 15.18s. Compared to the baseline results in Table 2, the absence of either component leads to a notable drop in accuracy. When removing SFT from DrugPilot, tool-selection accuracy drops to 92.9% (–5.8%) and parameter-extraction accuracy drops to 59.2% (–36.9%), with latency increasing by 16.45s. Meanwhile, removing Fe-Fo causes tool-selection accuracy to fall to 91.3% (–7.4%) and parameter-extraction accuracy to 88.3% (–7.8%), while latency increases by 2.32s. Specifically, SFT effectively enhances the model's understanding of domain-specific text, while the Fe-Fo mechanism significantly improves the agent's error-handling capability with minimal latency cost.

Effect of PMP: This study highlights the introduction of PMP (Parameterized Memory Pool), designed to handle large-scale molecular data. To validate the effectiveness of PMP, we gradually increased the number of molecules to be processed in a single query and tested the boundary of parameter recognition and extraction capability for different agent methods. To simulate real-world usage, an equivalent number of molecules were loaded into the memory pool. Under the same test data and model settings, we recorded the accuracy and latency of various agents. As shown in Figure 8a and Figure 8b, the performance of traditional approaches degrades as the number of molecules increases—accuracy drops continuously while latency rises sharply. In particular, when the number of molecules exceeds 15, other methods' accuracy drops significantly. When the number of molecules reaches 20, the CoT method essentially fails to extract valid

parameters. Similarly, DrugPilot without PMP, as well as ReAct and LoT, become nonfunctional when handling around 30 molecules. In contrast, real-world applications often require processing tens of thousands of molecules at once. DrugPilot equipped with PMP, however, maintains stable accuracy and response time throughout. The incorporation of PMP enables the model to focus on understanding user intent, effectively overcoming the challenge of extracting task-relevant parameters from lengthy and complex molecular descriptions.

6   Conclusion

To bridge the gap between AI technologies and practical drug discovery applications, we developed DrugPilot, an LLM-based parameterized reasoning agent to automate multi-stage drug discovery workflows. The DrugPilot achieves end-to-end automation of complex, multi-stage drug discovery workflows while overcoming critical field challenges. A core component of DrugPilot is its parameterized memory pool, which enables seamless processing of multi-modal drug data by parameterized representations, such as text, molecular graphs, and files. This design theoretically supports infinite sample processing capacity, effectively bypassing context length limitations commonly associated with LLMs. Furthermore, we introduce a novel feedback-focus mechanism that ensures robust reasoning accuracy and task consistency during prolonged operations. This mechanism demonstrates superior performance across multi-task coordination and cross-modal reasoning tasks. Additionally, we constructed the TCDD, a domain-specific dataset across 8 essential drug discovery tasks for model fine-tuning and evaluation. On the TCDD dataset, DrugPilot achieves task completion rates of 98.0%, 93.5%, and 64.0% on simple, multiple, and multi-turn tasks, respectively, substantially surpassing existing SOTA methods. Our work demonstrates how LLM-driven agents can transform drug discovery into an efficient, automated process, benefiting both AI development and pharmaceutical research.

Author contributions

Kun Li: Conceptualization, Methodology, Writing - Original Draft, Visualization, Supervision. Zhennan Wu: Methodology, Data curation, Writing - Original Draft. Shoupeng Wang: Data curation, Validation, Writing - Original Draft. Wenbin Hu: Project administration, Writing- Reviewing and Editing, Funding acquisition.

References

[1] Mingquan Liu, Chunyan Li, Ruizhe Chen, Dongsheng Cao, and Xiangxiang Zeng. Geometric deep learning for drug discovery. *Expert Systems with Applications*, 240:122498, 2024.

[2] Denise B Catacutan, Jeremie Alexander, Autumn Arnold, and Jonathan M Stokes. Machine learning in preclinical drug discovery. *Nature Chemical Biology*, 20(8):960–973, 2024.

[3] Yan A Ivanenkov, Daniil Polykovskiy, Dmitry Bezrukov, Bogdan Zagribelnyy, Vladimir Aladinskiy, Petrina Kamya, Alex Aliper, Feng Ren, and Alex Zhavoronkov. Chemistry42: an ai-driven platform for molecular design and optimization. *Journal of chemical information and modeling*, 63(3):695–701, 2023.

[4] Keda Yang, Zewen Xie, Zhen Li, Xiaoliang Qian, Nannan Sun, Tao He, Zuodong Xu, Jing Jiang, Qi Mei, Jie Wang, et al. Molprophet: a one-stop, general purpose, and ai-based platform for the early stages of drug discovery. *Journal of Chemical Information and Modeling*, 64(8):2941–2947, 2024.

[5] Chao Shen, Jianfei Song, Chang-Yu Hsieh, Dongsheng Cao, Yu Kang, Wenling Ye, Zhenxing Wu, Jike Wang, Odin Zhang, Xujun Zhang, et al. Drugflow: an ai-driven one-stop platform for innovative drug discovery. *Journal of Chemical Information and Modeling*, 64(14):5381–5391, 2024.

[6] François-Xavier Blaudin de Thé, Claire Baudier, Renan Andrade Pereira, Céline Lefebvre, Philippe Moingeon, Patrimony Working Group, et al. Transforming drug discovery with a high-throughput ai-powered platform: A 5-year experience with patrimony. *Drug Discovery Today*, 28(11):103772, 2023.

[7] Prafulla C Tiwari, Rishi Pal, Manju J Chaudhary, and Rajendra Nath. Artificial intelligence revolutionizing drug development: Exploring opportunities and challenges. *Drug Development Research*, 84(8):1652–1663, 2023.

[8] Javier Sánchez Lorente, Aleksandr V Sokolov, Gavin Ferguson, Helgi B Schiöth, Alexander S Hauser, and David E Gloriam. Gpcr drug discovery: new agents, targets and indications. *Nature Reviews Drug Discovery*, pages 1–22, 2025.

[9] Yuanqi Du, Arian R Jamasb, Jeff Guo, Tianfan Fu, Charles Harris, Yingheng Wang, Chenru Duan, Pietro Liò, Philippe Schwaller, and Tom L Blundell. Machine learning-aided generative molecular design. *Nature Machine Intelligence*, 6(6):589–604, 2024.

[10] Hongjie Wu, Junkai Liu, Tengsheng Jiang, Quan Zou, Shujie Qi, Zhiming Cui, Prayag Tiwari, and Yijie Ding. Attentionmgt-dta: A multi-modal drug-target affinity prediction using graph transformer and attention mechanism. *Neural Networks*, 169:623–636, 2024.

[11] Mukun Chen, Jia Wu, Shirui Pan, Fu Lin, Bo Du, Xiuwen Gong, and Wenbin Hu. Knowledge-aware contrastive heterogeneous molecular graph learning. *arXiv preprint arXiv:2502.11711*, 2025.

[12] Kun Li, Longtao Hu, Xiantao Cai, Jia Wu, and Wenbin Hu. Can molecular evolution mechanism enhance molecular representation? *arXiv preprint arXiv:2501.15799*, 2025.

[13] Anastasiia V Sadybekov and Vsevolod Katritch. Computational approaches streamlining drug discovery. *Nature*, 616(7958):673–685, 2023.

[14] Kit-Kay Mak and Mallikarjuna Rao Pichika. Artificial intelligence in drug development: present status and future prospects. *Drug discovery today*, 24(3):773–780, 2019.

[15] Sarfaraz K Niazi and Zamara Mariam. Computer-aided drug design and drug discovery: a prospective analysis. *Pharmaceuticals*, 17(1):22, 2023.

[16] Zhaoyi Chen, Xiong Liu, William Hogan, Elizabeth Shenkman, and Jiang Bian. Applications of artificial intelligence in drug development using real-world data. *Drug discovery today*, 26(5):1256–1264, 2021.

[17] Alex Zhavoronkov, Yan A Ivanenkov, Alex Aliper, Mark S Veselov, Vladimir A Aladinskiy, Anastasiya V Aladinskaya, Victor A Terentiev, Daniil A Polykovskiy, Maksim D Kuznetsov, Arip Asadulaev, et al. Deep learning enables rapid identification of potent ddr1 kinase inhibitors. *Nature biotechnology*, 37(9):1038–1040, 2019.

[18] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

[19] Feng Ren, Alex Aliper, Jian Chen, Heng Zhao, Sujata Rao, Christoph Kuppe, Ivan V Ozerov, Man Zhang, Klaus Witte, Chris Kruse, et al. A small-molecule tnik inhibitor targets fibrosis in preclinical and clinical models. *Nature Biotechnology*, 43(1):63–75, 2025.

[20] Kang Zhang, Xin Yang, Yifei Wang, Yunfang Yu, Niu Huang, Gen Li, Xiaokun Li, Joseph C Wu, and Shengyong Yang. Artificial intelligence in drug development. *Nature Medicine*, pages 1–15, 2025.

[21] Chiranjib Chakraborty, Manojit Bhattacharya, Soumen Pal, Srijan Chatterjee, Arpita Das, and Sang-Soo Lee. Ai-enabled language models (lms) to large language models (llms) and multimodal large language models (mllms) in drug discovery and development. *Journal of Advanced Research*, 2025.

[22] Geyan Ye, Xibao Cai, Houtim Lai, Xing Wang, Junhong Huang, Longyue Wang, Wei Liu, and Xiangxiang Zeng. Drugassist: a large language model for molecule optimization. *Briefings in Bioinformatics*, 26(1):bbae693, 01 2025.

[23] Raghad J. AbuNasser, Mostafa Z. Ali, Yaser Jararweh, Mustafa Daraghmeh, and Talal Z. Ali. Large language models in drug discovery: A comprehensive analysis of drug-target interaction prediction. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 417–431, 2024.

[24] Sizhe Liu, Yizhou Lu, Siyu Chen, Xiyang Hu, Jieyu Zhao, Tianfan Fu, and Yue Zhao. Drugagent: Automating ai-aided drug discovery programming through llm multi-agent collaboration. In *2nd AI4Research Workshop: Towards a Knowledge-grounded Scientific Research Lifecycle*.

[25] Yoshitaka Inoue, Tianci Song, Xinling Wang, Augustin Luna, and Tianfan Fu. Drugagent: Multi-agent large language model-based reasoning for drug-target interaction prediction. In *ICLR 2025 Workshop on Machine Learning for Genomics Explorations*, 2025.

[26] Yizhen Zheng, Huan Yee Koh, Jiaxin Ju, Anh T. N. Nguyen, Lauren T. May, Geoffrey I. Webb, and Shirui Pan. Large language models for scientific discovery in molecular property prediction. *Nature Machine Intelligence*, 7(3):437–447, Mar 2025.

[27] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

[28] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[29] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[30] Tongxuan Liu, Wenjiang Xu, Weizhe Huang, Yuting Zeng, Jiaxing Wang, Xingyu Wang, Hailong Yang, and Jing Li. Logic-of-thought: Injecting logic into contexts for full reasoning in large language models. *arXiv preprint arXiv:2409.17539*, 2024.

[31] Geyan Ye, Xibao Cai, Houtim Lai, Xing Wang, Junhong Huang, Longyue Wang, Wei Liu, and Xiangxiang Zeng. Drugassist: A large language model for molecule optimization. *Briefings in Bioinformatics*, 26(1):bbae693, 2025.

[32] Haorui Wang, Marta Skreta, Cher Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, Yuanqi Du, Alan Aspuru-Guzik, Kirill Neklyudov, and Chao Zhang. Efficient evolutionary search over chemical space with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[33] Xianggen Liu, Yan Guo, Haoran Li, Jin Liu, Shudong Huang, Bowen Ke, and Jiancheng Lv. Drugllm: Open large language model for few-shot molecule generation. *arXiv preprint arXiv:2405.06690*, 2024.

[34] Yuyan Liu, Sirui Ding, Sheng Zhou, Wenqi Fan, and Qiaoyu Tan. Moleculargpt: Open large language model (llm) for few-shot molecular property prediction. *arXiv preprint arXiv:2406.12950*, 2024.

[35] Juan Manuel Zambrano Chaves, Eric Wang, Tao Tu, Eeshit Dhaval Vaishnav, Byron Lee, S Sara Mahdavi, Christopher Semturs, David Fleet, Vivek Natarajan, and Shekoofeh Azizi. Tx-llm: A large language model for therapeutics. *arXiv preprint arXiv:2406.06316*, 2024.

[36] Shiyu Fang, Jiaqi Liu, Mingyu Ding, Yiming Cui, Chen Lv, Peng Hang, and Jian Sun. Towards interactive and learnable cooperative driving automation: a large language model-driven decision-making framework. *IEEE Transactions on Vehicular Technology*, 2025.

[37] Yuchen Xia, Jize Zhang, Nasser Jazdi, and Michael Weyrich. Incorporating large language models into production systems for enhanced task automation and flexibility. *arXiv preprint arXiv:2407.08550*, 2024.

[38] Jin Xiao, YiXiao Chen, LinFeng Zhang, Han Wang, and Tong Zhu. A machine learning-based high-precision density functional method for drug-like molecules. *Artificial Intelligence Chemistry*, 2(1):100037, 2024.

[39] Haiping Zhang, Hongjie Fan, Jixia Wang, Tao Hou, Konda Mani Saravanan, Wei Xia, Hei Wun Kan, Junxin Li, John ZH Zhang, Xinmiao Liang, et al. Revolutionizing gpcr–ligand predictions: Deepgpcr with experimental validation for high-precision drug discovery. *Briefings in Bioinformatics*, 25(4):bbae281, 2024.

[40] Taicheng Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh Chawla, Olaf Wiest, Xiangliang Zhang, et al. What can large language models do in chemistry? a comprehensive benchmark on eight tasks. *Advances in Neural Information Processing Systems*, 36:59662–59688, 2023.

[41] NAN SHAO, Zefan Cai, Hanwei xu, Chonghua Liao, Yanan Zheng, and Zhilin Yang. Compositional task representations for large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[42] Kaituo Feng, Changsheng Li, Xiaolu Zhang, Jun Zhou, Ye Yuan, and Guoren Wang. Keypoint-based progressive chain-of-thought distillation for LLMs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 13241–13255. PMLR, 21–27 Jul 2024.

[43] Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *Forty-first International Conference on Machine Learning*, 2024.

[44] Yizhen Zheng, Huan Yee Koh, Maddie Yang, Li Li, Lauren T May, Geoffrey I Webb, Shirui Pan, and George Church. Large language models in drug discovery and development: From disease mechanisms to clinical trials. *arXiv preprint arXiv:2409.04481*, 2024.

[45] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, et al. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):1–40, 2024.

[46] Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An llm compiler for parallel function calling. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[47] Weiwen Liu, Xu Huang, Xingshan Zeng, 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. ToolACE: Winning the points of LLM function calling. In *The Thirteenth International Conference on Learning Representations*, 2025.

[48] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690, 2024.

[49] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[50] Ariel Lee, Cole Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

[51] Yilun Kong, Jingqing Ruan, Yihong Chen, Bin Zhang, Tianpeng Bao, Shiwei Shi, Guoqing Du, Xiaoru Hu, Hangyu Mao, Ziyue Li, Xingyu Zeng, Rui Zhao, and Xueqian Wang. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world industry systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 371–385, Miami, Florida, US, November 2024. Association for Computational Linguistics.

[52] Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.

[53] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023.

[54] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. EXPEL: LLM agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2023.

[55] Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*, 2024.

[56] Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can context length of open-source llms truly promise? *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[57] Xinnian Liang, Bing Wang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. Unleashing infinite-length input capacity for large-scale language models with self-controlled memory system. *arXiv preprint arXiv:2304.13343*, 2023.

[58] Lei Wang, Jingsen Zhang, Hao Yang, Zhiyuan Chen, Jiakai Tang, Zeyu Zhang, Xu Chen, Yankai Lin, Ruihua Song, Wayne Xin Zhao, Jun Xu, Zhicheng Dou, Jun Wang, and Ji-Rong Wen. When large language model based agent meets user behavior analysis: A novel user simulation paradigm, 2023. Preprint.

[59] Peter J Denning. The locality principle. *Communications of the ACM*, 48(7):19–24, 2005.

[60] Wanjun Zhong, Lianghong Guo, Qiqi Gao, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023.

[61] Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. Ret-llm: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*, 2023.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[63] Joshua Holstein. Bridging domain expertise and ai through data understanding. In *Companion Proceedings of the 29th International Conference on Intelligent User Interfaces*, pages 163–165, 2024.

[64] Hao Zhu. Big data and artificial intelligence modeling for drug discovery. *Annual review of pharmacology and toxicology*, 60(1):573–589, 2020.

[65] Wanjuan Yang, Jorge Soares, Patricia Greninger, Elena J Edelman, Howard Lightfoot, Simon Forbes, Nidhi Bindal, Dave Beare, James A Smith, I Richard Thompson, et al. Genomics of drug sensitivity in cancer (gdsc): a resource for therapeutic biomarker discovery in cancer cells. *Nucleic acids research*, 41(D1):D955–D961, 2012.

[66] Mindy I Davis, Jeremy P Hunt, Sanna Herrgard, Pietro Ciceri, Lisa M Wodicka, Gabriel Pallares, Michael Hocker, Daniel K Treiber, and Patrick P Zarrinkar. Comprehensive analysis of kinase inhibitor selectivity. *Nature biotechnology*, 29(11):1046–1051, 2011.

[67] Jing Tang, Agnieszka Szwajda, Sushil Shakyawar, Tao Xu, Petteri Hintsanen, Krister Wennerberg, and Tero Aittokallio. Making sense of large-scale kinase inhibitor bioactivity data sets: a comparative and integrative analysis. *Journal of chemical information and modeling*, 54(3):735–743, 2014.

[68] Tiqing Liu, Yuhmei Lin, Xin Wen, Robert N Jorissen, and Michael K Gilson. Bindingdb: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic acids research*, 35(suppl_1):D198–D201, 2007.

[69] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018.

[70] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

[71] Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

[72] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[73] Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

[74] Mirac Suzgun, Luke Melas-Kyriazi, Suproteem Sarkar, Scott D Kominers, and Stuart Shieber. The harvard uspto patent dataset: A large-scale, well-structured, and multi-purpose corpus of patent applications. *Advances in neural information processing systems*, 36:57908–57946, 2023.

[75] Raja Vavekanand and Kira Sam. Llama 3.1: An in-depth analysis of the next-generation large language model, 2024.

[76] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu,

Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

[77] Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, et al. Llm pruning and distillation in practice: The minitron approach. *arXiv preprint arXiv:2408.11796*, 2024.

[78] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[79] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report. *CoRR*, abs/2407.10671, 2024.

[80] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, Alex X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. Deepseek LLM: scaling open-source language models with longtermism. *CoRR*, abs/2401.02954, 2024.

[81] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025.

[82] OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, and Alex Tachard Passos. Gpt-4o: A multimodal large language model. *arXiv*, 2024.

A Memory Pool Prompt

```
┌──  Memory Pool Prompt ──────────────────────────────────────┐
# describe responsibilities
You have access to a memory pool. You are responsible for passing in the correct arguments
to the tool you choose to use.

# Define the input format
Each input will consist of two parts:
1. Question description or Observation 2. What parameters are in the memory pool.

# explain how LLM uses memory pool
# when not to use memory pool
You should first try to extract the argument that this tool need from the problem
description, such as 'drug_smiles': 'C=C1C2=CCCC=C(NC(C)'.
# when and how to use memory pool
If the question description does not contain this argumennt, then you need to find this
argument in the memory pool, using '(key in the MemoryPool)' to indicate this choice.
You should use () to read MemoryPool.
# give an example
For example, if MemoryPool(arguments=dict_keys(['user_smiles', 'generated_smiles',
'optimized_smiles'])), then you can use 'drug_smiles': '(user_smiles)' or 'drug_smiles':
'(generated_smiles)' or 'drug_smiles': '(optimized_smiles) '.
# wrong example
You cannot use arguments that are not in the memory pool. For example, if MemoryPool does
not contain 'user_target_seq', you should not give 'target_seq': '(user_target_seq)'.
└──────────────────────────────────────────────────────────────┘
```

Figure 9: Memory pool prompt.

To help LLMs better understand the PMP in DrugPilot, we have incorporated a memory pool prompt into the system prompt. The full memory pool prompt is shown in Figure 9.

The memory pool prompt first clarifies the existence of PMP and the responsibilities of the LLMs, namely the correct transmission of parameters to the tools. It then defines the input format received by the LLMs, comprising two parts: the user's question or the tool's output, and a description of the current state of PMP, which includes the list of currently stored keys. LLMs can select a key from this pool and map it to its corresponding value. It then explains in detail how the LLMs should interact with PMP. First, it defines scenarios where PMP should not be used: if the required parameters are already present in the question, the LLMs should extract them directly. Next, it specifies when and how to use PMP: if the question lacks the necessary parameters, the LLMs must retrieve the corresponding key from the memory pool and enclose it in parentheses to indicate retrieval. Finally, the prompt provides both a correct and an incorrect example, demonstrating proper memory pool usage and helping LLMs avoid retrieving non-existent keys, thereby mitigating hallucination.

B Reasoning errors of LLM

In tool calling, LLMs are required to generate an action input in JSON format, containing the tool name to be called and required parameters. And in actual tasks, there will be frequent interactions with PMP. Therefore, problems will inevitably arise both in content and format. Based on the real output of LLMs, we summarized the common reasoning errors as shown in Figure 10.

C Fine-Tuning Configuration

We conducted LoRA fine-tuning on the LLMs used in DrugPilot to enhance their domain knowledge in drug discovery and improve their ability to call drug-related tools. A batch size of 4 was used for smaller models, and 8 for larger ones.
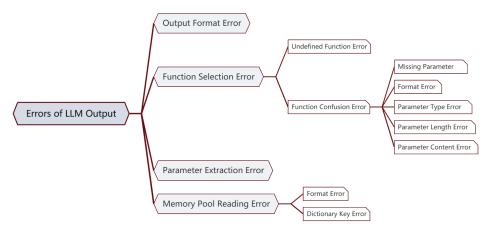
Figure 10: Common reasoning errors of LLMs. The common types of reasoning errors when LLMs call drug-related tools, and the Fe-Fo mechanism will provide feedback to LLMs regarding these issues.

We deployed the final inference-stage LLMs on the Ollama[4] platform. The hyperparameter settings used during the fine-tuning process are detailed in Table 5.

Hyperparameter	Value / Strategy
Batch size	4-8
Cutoff length	1024
Optimizer	AdamW
Initial learning rate	5e-5
Learning rate scheduler	Cosine decay
Precision	BF16
Number of epochs	3
Deployment platform	Ollama

Table 5: Hyperparameter Settings for Fine-tuning

D Accuracy Calculation

Acc.F and Acc.P represent the accuracy of tool selection and parameter extraction, and they are defined by:

$$Acc.F = \frac{1}{N_c} \sum_{i=1}^{N_c} F(sample_i)$$

$$Acc.P = \frac{1}{N_c} \sum_{i=1}^{N_c} P(sample_i)$$

where N_c is the number of samples in a category, $F(\cdot)$ and $P(\cdot)$ are indicator functions denote whether the current sample has correctly selected the function and extracted the parameters.
