Appendix 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)'.
```

Fig. A1: 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 Fig. A1.

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.

Appendix B DrugPilot's Reasoning Errors

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

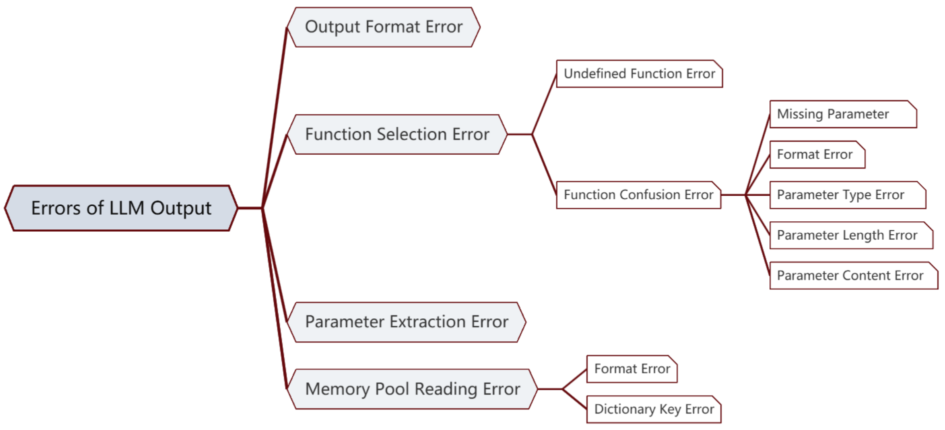

Fig. B2: Common reasoning errors of LLMs. The common types of reasoning errors when LLMs call drug-related tools, and the Fe-Fo mechanism will provide feedback to LLMs regarding these issues.

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 C1: Hyperparameter Settings for Fine-tuning

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 Fig.B2.

Appendix 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. Batch size of 4 was used for smaller models, and 8 for larger ones. We deployed the final inference-stage LLMs on the Ollama[1] platform. The hyperparameter settings used during the fine-tuning process are detailed in Table C1.

[1] https://github.com/ollama/ollama.

Datasets	Metrics	Molecule Generation Task			
		CDGS [59]	GruM-2D [60]	MOOD [61]	RMCD (Ours)
QM9 +	FCD ↓	77.0 / 61.1 / 53.1	85.3 / 60.7 / 52.7	80.2 / 57.7 / 48.7	**77.0 / 56.0 / 47.8**
GDSCv2	MMD ↓	.340 / .142 / .110	.337 / .138 / .106	.347 / .195 / .144	**.313 / .142 / .101**

Datasets	Metrics	Molecular Optimization Task			
		Prompt-MolOpt [62]	HN-GFN [63]	FFLOM [64]	FMOP (Ours)
QM9 +	Success ↑	91.68%	92.70%	88.83%	**95.43%**
GDSCv2	Improv. ↑	5.70%	3.30%	6.3%	**7.50%**

Datasets	Metrics	Drug Target Interaction Prediction Task			
		FOTF-CPI [65]	HiGraphDTI [66]	MGNDTI [67]	SiamDTI (Ours)
BindingDB	AUROC ↑	0.506 ± 0.030	0.540 ± 0.030	0.524 ± 0.032	**0.554 ± 0.016**
BioSNAP		0.590 ± 0.030	0.629 ± 0.030	0.601 ± 0.012	**0.636 ± 0.020**

Datasets	Metrics	Drug Target Affinity Prediction Task			
		RF [68]	MSGNN-DTA [69]	HGRL-DTA [70]	CLG-DTA (Ours)
KIBA	PCC ↑	0.150	0.116	0.083	**0.280**
	MSE ↓	0.962	0.907	1.024	**0.900**

Datasets	Metrics	Molecular Property Prediction Task			
		KCL [71]	GROVER [72]	CoMPT [73]	KCHML (Ours)
QM7	MSE ↓	59.9 ± 2.8	90 ± 1.9	86.5 ± 1.3	**56.1 ± 3.5**
QM8		0.0130 ± 0.013	0.0180 ± 0.001	0.0187 ± 0.001	**0.0121 ± 0.000**
ESOL	RMSE ↓	0.659 ± 0.019	1.435 ± 0.283	0.832 ± 0.039	**0.612 ± 0.142**
FreeSolv		1.148 ± 0.257	2.935 ± 0.620	1.940 ± 0.808	**1.136 ± 0.142**
Lipo		0.566 ± 0.007	0.829 ± 0.010	0.647 ± 0.028	**0.527 ± 0.009**
BACE	ROC-AUC ↑	93.00 ± 0.69	82.34 ± 8.83	82.47 ± 0.69	**94.57 ± 1.63**
BBBP		95.38 ± 1.70	84.37 ± 4.10	94.57 ± 1.20	**95.89 ± 1.66**

Datasets	Metrics	Drug-Drug Interaction Prediction Task			
		GMPNN [74]	DGNN-DDI [75]	DSN-DDI [76]	KCHML (Ours)
TwoSide[a]	AUC ↑	77.69 ± 0.26	77.25 ± 0.23	77.25 ± 0.23	**81.87 ± 0.54**
TwoSide[b]		80.91 ± 0.80	81.96 ± 0.26	81.59 ± 0.66	**83.75 ± 0.89**

Datasets	Metrics	Drug Response Prediction Task[c]			
		$\text{MSDA}_{\text{GraTransDRP}}$	$\text{MSDA}_{\text{TransEDRP}}$	$\text{CLDR}_{\text{GraTransDRP}}$	$\text{CLDR}_{\text{TransEDRP}}$
GDSCv2	PCC ↑	0.5103 (5.3%)	0.5316 (+5.1%)	**0.5288 (+11.61%)**	0.5149 (+1.77%)
	MSE ↓	0.0039 (+20.6%)	0.0044 (15.2%)	0.0039 (+17.02%)	**0.0038 (+26.23%)**

Datasets	Metrics	Drug Retrosynthesis Task			
		GET-LT1 [77]	SCROP [78]	RetroXpert [79]	CFC-Retro (Ours)
USPTO-50K	Top-1 ↑	59.1	59.0	62.1	**65.9**
	Top-3 ↑	73.4	74.8	75.8	**80.4**
	Top-5 ↑	76.4	78.1	78.5	**82.4**

[a]: Both molecules unseen in training data. [b]: Only one molecule present in training.
[c]: MSDA and CLDR are enhancement plug-ins introduced by Ours to improve the performance of baseline methods such as GraTransDRP [80] and TransEDRP [81].

Table C2: Performance comparison across task-specific AI model zoo. The results demonstrate how our methods perform across different tasks, in comparison with the current SOTA methods.

Appendix D AI Model Zoo

DrugPilot is designed to facilitate the accurate and efficient execution of multiple tasks within the drug research and development pipeline by leveraging tool calling mechanisms. For each specific task, there are various models that are tailored to complete the task. Table C2 presents a comprehensive comparison of the performance achieved by our method against a range of SOTA baselines. Our method consistently achieves superior performance across all tasks, which ensures the reliability and effectiveness of individual task execution. Consequently, this contributes to the enhanced robustness of the overall DrugPilot workflow.