SMART: Self-Aware Agent for Tool Overuse Mitigation

Cheng Qian[1*], **Emre Can Acikgoz**[1*], **Hongru Wang**[1†], **Xiusi Chen**[1], **Avirup Sil**[2], **Dilek Hakkani-Tür**[1], **Gokhan Tur**[1], **Heng Ji**[1†]
[1]University of Illinois Urbana-Champaign, [2]IBM Research AI
{chengq9, acikgoz2, hengji}@illinois.edu

Abstract

Current Large Language Model (LLM) agents demonstrate strong reasoning and tool use capabilities, but often lack self-awareness, failing to balance these approaches effectively. This imbalance leads to **Tool Overuse**, where models unnecessarily rely on external tools for tasks solvable with parametric knowledge, increasing computational overhead. Inspired by human metacognition, we introduce **SMART** (Strategic Model-Aware Reasoning with Tools), a paradigm that enhances an agent's self-awareness to optimize task handling and reduce tool overuse. To support this paradigm, we introduce **SMART-ER**, a dataset spanning three domains, where reasoning alternates between parametric knowledge and tool-dependent steps, with each step enriched by rationales explaining when tools are necessary. Through supervised training, we develop **SMARTAgent**, a family of models that dynamically balance parametric knowledge and tool use. Evaluations show that SMARTAgent reduces tool use by 24% while improving performance by over 37%, enabling 7B-scale models to match its 70B counterpart and GPT-4o. Additionally, SMARTAgent generalizes to out-of-distribution test data like GSM8K and MINTQA, maintaining accuracy with just one-fifth the tool calls. These highlight the potential of strategic tool use to enhance reasoning, mitigate overuse, and bridge the gap between model size and performance, advancing intelligent and resource-efficient agent designs. All the data and codes are released[1].

1 Introduction

Recent advancements in Large Language Models (LLMs) (Ouyang et al., 2022; Team et al., 2023; Dubey et al., 2024) have led to remarkable improvements in reasoning capabilities, driving progress

[*] Indicates equal contribution.
[†] Mentorship
[1] https://github.com/qiancheng0/Open-SMARTAgent

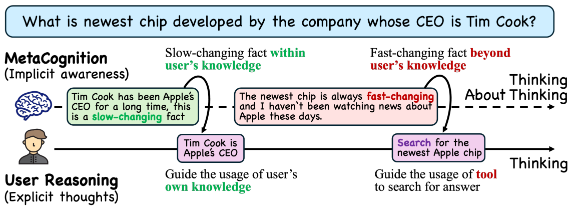

What is newest chip developed by the company whose CEO is Tim Cook?
MetaCognition
(Implicit awareness)
Slow-changing fact within user's knowledge
Fast-changing fact beyond user's knowledge
Thinking About Thinking
Tim Cook has been Apple's CEO for a long time, this is a slow-changing fact
The newest chip is always fast-changing and I haven't been watching news about Apple these days.
Tim Cook is Apple's CEO
Search for the newest Apple chip
Thinking
User Reasoning
(Explicit thoughts)
Guide the usage of user's own knowledge
Guide the usage of tool to search for answer

Figure 1: An illustration of human metacognition: The user recalls Tim Cook's role from **prior knowledge** (a *slow-changing fact*), but uses **online search** to find the latest chip info (a *fast-changing fact*).

in diverse domains such as coherent text composition (Wei et al., 2022a), code generation (Gao et al., 2023; Wang et al., 2025b; Pan et al., 2024), complex logical deduction (Yao et al., 2023, 2024), and nuanced natural language understanding (Wang et al., 2023; Yu et al., 2024; Wu et al., 2025). However, challenges remain, such as the inability to handle real-time information (Yu and Ji, 2024), model real-world challenges (Qian et al., 2025b), provide accurate mathematical results (Lu et al., 2022), and fully comprehend human intentions (Qian et al., 2024b). These limitations highlight the need for LLMs to leverage external tools (Schick et al., 2023; Qin et al., 2023; Yuan et al., 2024; Qian et al., 2024a), enabling them to function as agents capable of assisting users in diverse tasks (Qin et al., 2024; Xi et al., 2023). Effective tool use and reasoning are thus complementary, each enhancing the other to overcome current shortcomings.

Therefore, in problem-solving, a language agent often combines reasoning with tool use, following a ReACT-style approach (Yao et al., 2023), where the model alternates between thought processes and actions to derive solutions. This enables the core agent to apply its parametric knowledge to advance task-solving while using external tools to address its limitations. However, this interplay raises a critical question: **when should the agent rely on external tools versus its own knowledge?**

To investigate this, we first conduct a preliminary study on both LLMs and LM-driven agent systems to assess their ability to dynamically and effectively switch between external tool use and parametric knowledge-driven reasoning. Our empirical results reveal a consistent bias, with LLMs unnecessarily invoking tools over 30% of the time, and agent systems exhibiting similar behavior even when their parametric knowledge alone would suffice. We identify this phenomenon as **Tool Overuse**, which arises from the model's inability to recognize when its internal knowledge is sufficient. This not only leads to unnecessary resource consumption but can also confuse the model, ultimately degrading performance. This observation highlights the need for better calibration of an agent's self-awareness, ensuring it can discern when to rely on tools versus its own knowledge. Striking this balance is crucial for enhancing efficiency, scalability, and user experience as LM-driven agents are increasingly deployed in real-world applications.

To address this challenge, we propose **SMART** (Strategic Model-Aware Reasoning with Tools), which draws inspiration from human decision-making to calibrate self-awareness in agent models for effective tool use and reasoning. In **Metacognitive Theory** (Schraw and Moshman, 1995), psychology highlights humans' awareness of their thought processes, including when to apply specific problem-solving strategies (Livingston, 2003). As Figure 1 illustrates, this implicit heuristic allows dynamic balancing between external strategies and internal knowledge (Minsky, 1986). Similarly, agents need metacognition to optimize tool usage. By aligning the model's subjective perception with its knowledge boundary, we enable agents to make more informed decisions on when to rely on external tools or internal knowledge.

We adopt a data-driven approach to calibrate model decision-making by constructing **SMART-ER** (SMART-Enhanced Reasoning), a dataset spanning three domains—Math, Time, and Intention. It addresses key LLM limitations, including computational accuracy (Hendrycks et al., 2021), outdated knowledge (Vu et al., 2023), and user preference awareness (Qian et al., 2024b). Specifically, each question in SMART-ER combines sub-questions the model handles well (e.g., simple arithmetic, static facts, commonsense) with those it struggles with (e.g., complex math, dynamic facts, user-specific intentions). We break down each question into reasoning steps, categorizing them as either parametric knowledge-driven or tool-dependent. For parametric steps, we provide reasoning based on internal knowledge. For tool-dependent steps, we map them to appropriate tools, execute them, and integrate the results into the reasoning process. Finally, inspired by metacognitive heuristics, we refine each step with explicit justifications, clarifying when parametric knowledge suffices or external tools are needed. By transforming implicit decision-making heuristics into explicit language-based reasoning, we guide the model to develop calibrated awareness of its knowledge boundaries.

Leveraging SMART-ER, we develop **SMARTAgent**, a family of agent models designed to dynamically balance reasoning between parametric knowledge and external tools. Empirical results show that SMARTAgent reduces tool use by 24% while improving overall performance by over 37%, effectively mitigating tool overuse. Notably, it enables 7B-scale models to match the performance of GPT-4 and 70B models, bridging the gap between model size and capability. Additionally, SMARTAgent efficiently handles out-of-distribution (OOD) tasks, requiring only one-fifth the number of tool calls while preserving accuracy. Finally, analysis of SMARTAgent's confidence through logits reveals more certain reasoning-tool-switching decisions, further validating our approach in calibrating the agent's self-awareness. In summary:

- We identify and define the issue of **Tool Overuse**, emphasizing that strategically balancing the complementary strengths of knowledge-driven reasoning and external tool calls can mitigate this problem in both LLMs and agent systems.
- We introduce **SMART-ER**, a multi-domain dataset designed to address key limitations of agent models by integrating metacognitive heuristics to better help them recognize and adapt to their knowledge boundaries.
- We develop **SMARTAgent**, a family of agents that intelligently balances parametric reasoning and tool use, achieving improved performance, reduced tool overuse, and more confident decision-making in tool utilization.

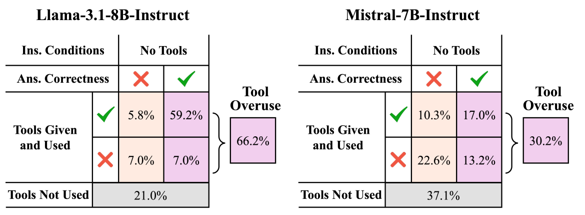
2 Related Work

LM Knowledge Boundary. Recent studies highlight that while LMs excel at standard tasks, they struggle to recognize and acknowledge the limits of their knowledge (Yin et al., 2023; Qian et al., 2023b; Kadavath et al., 2022). To address this gap, the concept of knowledge boundary has been introduced to define the limits of knowledge in LLMs (Li et al., 2024; Amayuelas et al., 2023). Building on this, some research evaluates LMs' self-awareness of their knowledge boundary through verbal probing (Kadavath et al., 2022) and fine-grained benchmarks (Yin et al., 2024), enabling LMs to determine whether a question is answerable. Other work focuses on mitigating hallucinations arising from the model's unawareness of its limits through data augmentation (Chen et al., 2023, 2024b), retrieval augmentation (Ren et al., 2023), and confidence calibration (Xue et al., 2024). Additionally, Chen et al. (2024a) and Zhang et al. (2024) trained LLMs to express their knowledge boundaries, enabling them to answer known questions and admit ignorance for unknown ones. Recently, reinforcement learning has been increasingly explored as a means to help models recognize their knowledge boundaries to guide more efficient decisions (Qian et al., 2025a; Wang et al., 2025a). Our work aligns with these studies and focuses on enhancing agents' awareness for wiser tool use.

LM Tool Use. Integrating tool use into LLMs has gained significant attention as a way to complement parametric knowledge and enhance decision-making (Qin et al., 2023; Qu et al., 2025). Some research focuses on enabling LLMs to access external tools to overcome knowledge limitations (Qin et al., 2024; Qian et al., 2024d), including up-to-date information (Vu et al., 2023; Wang et al., 2024b) and domain-specific expertise (Ling et al., 2023; Wang et al., 2024a). Others explore tool creation (Qian et al., 2023a; Cai et al., 2024) and external module integration (Qian et al., 2024c) to improve tool learning robustness. Despite these, a key challenge lies in evaluating and enhancing LLMs' ability to determine when and which tools to use. Benchmarks like MetaTool (Huang et al., 2023) and WTU-EVAL (Ning et al., 2024) highlight LLMs' struggles with unnecessary or incorrect tool usage, while dynamic frameworks (Wang et al., 2024c; Shen et al., 2024) propose adaptively invoking tools based on internal uncertainty thresholds. Unlike prior works, SMART rigorously defines *tool overuse* and addresses it by optimizing the balance of internal knowledge and tool use.

3 Preliminaries

To investigate how models decide between invoking tools and relying on their own knowledge, we conduct a preliminary study on both LLMs and LM-driven agent systems. Our findings reveal both LLMs and agent systems' strong tendency for excessive tool use, which we define as **Tool Overuse**, leading to unnecessary resource overhead.



Llama-3.1-8B-Instruct

Ins. Conditions		No Tools	
Ans. Correctness		✗	✓
Tools Given and Used	✓	5.8%	59.2%
	✗	7.0%	7.0%
Tools Not Used		21.0%	

Tool Overuse: 66.2%

Mistral-7B-Instruct

Ins. Conditions		No Tools	
Ans. Correctness		✗	✓
Tools Given and Used	✓	10.3%	17.0%
	✗	22.6%	13.2%
Tools Not Used		37.1%	

Tool Overuse: 30.2%

Figure 2: Statistics on Llama and Mistral's tool overuse.

Tools / Envs	Python Code	File System	User Interface
XAgent	7	35	2
AgentGPT	16	-	-

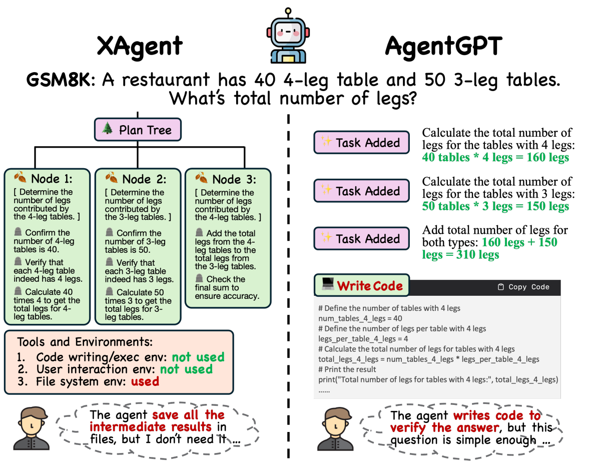
Table 1: Statistics on XAgent and AgentGPT's tool overuse. Both agents invoke tools multiple times across 50 samples, despite *ideally* requiring *zero* tool usage.

Definition of Tool Overuse. Tool overuse refers to the excessive reliance on external tools when an agent model could have successfully completed the task using its parametric knowledge alone. Formally, let Q be the total set of questions, and let P be the subset of questions that the model can correctly answer without using any tools. The model's intrinsic reasoning capability is then given by $\alpha = \frac{|P|}{|Q|}$. Now, suppose that when provided with access to tools, the model chooses to invoke at least one tool on a fraction β of these questions in P. The **Tool Overuse Rate** is then defined as:

$$\mathcal{O} = \alpha \cdot \beta$$

which quantifies the proportion of all questions where tool use is unnecessary, highlighting inefficiencies in the model's decision-making process.

Experiments on LLMs. We first experiment with Llama-3.1-8B (Dubey et al., 2024) and Mistral-7B (Jiang et al., 2023) on the GSM8K test set (Cobbe et al., 2021). Each test question is presented under two conditions: i) the model reasons through the question normally and provides a final answer without using tools, and ii) the model has access to tools and independently decides whether to use them (see Appendix A.2). The statistics in Figure 2 reveal two key insights. First, both models exhibit significant tool overuse, with Llama's rate exceeding 50%. Second, in some cases, tool use leads to incorrect answers, even for questions the model could have solved correctly without external


XAgent
AgentGPT
GSM8K: A restaurant has 40 4-leg table and 50 3-leg tables. What's total number of legs?
Plan Tree
Node 1: [Determine the number of legs contributed by the 4-leg tables.]
Confirm the number of 4-leg tables is 40.
Verify that each 4-leg table indeed has 4 legs.
Calculate 40 times 4 to get the total legs for 4-leg tables.
Node 2: [Determine the number of legs contributed by the 3-leg tables.]
Confirm the number of 3-leg tables is 50.
Verify that each 3-leg table indeed has 3 legs.
Calculate 50 times 3 to get the total legs for 3-leg tables.
Node 3: [Determine the number of legs contributed by the 4-leg tables.]
Add the total legs from the 4-leg tables to the total legs from the 3-leg tables.
Check the final sum to ensure accuracy.
Tools and Environments:
1. Code writing/exec env: not used
2. User interaction env: not used
3. File system env: used
The agent save all the intermediate results in files, but I don't need it ...
Task Added
Calculate the total number of legs for the tables with 4 legs: 40 tables * 4 legs = 160 legs
Task Added
Calculate the total number of legs for the tables with 3 legs: 50 tables * 3 legs = 150 legs
Task Added
Add total number of legs for both types: 160 legs + 150 legs = 310 legs
Write Code
Copy Code
The agent writes code to verify the answer, but this question is simple enough ...

```
# Define the number of tables with 4 legs
num_tables_4_legs = 40
# Define the number of legs per table with 4 legs
legs_per_table_4_legs = 4
# Calculate the total number of legs for tables with 4 legs
total_legs_4_legs = num_tables_4_legs * legs_per_table_4_legs
# Print the result
print("Total number of legs for tables with 4 legs:", total_legs_4_legs)
......
```

Figure 3: Example cases on XAgent and AgentGPT's tool overuse.

assistance. This highlights how excessive reliance on tools can introduce unnecessary complexity and degrade performance.

Experiments on LM-driven Agents. In addition to LLMs, we also experiment with two agent systems: XAgent (Team, 2023) and AgentGPT (Team, 2024), both designed for complex problem-solving and driven by closed-source GPT models. We sampled 50 queries from the GSM8K test set that can be answered correctly without tools (see Appendix A.1) and instructed the models to use tools only when necessary. The results in Table 1 show that, despite being equipped with various tools, both agent systems still tend to use them unnecessarily, significantly slowing down problem-solving (about 10x slower than using GPT alone). We further provide a case study in Appendix A.1 highlighting issues such as XAgent redundantly saving results to files and AgentGPT unnecessarily invoking a code-writing tool after generating an answer. These observations underscore the need to address our core research question: **How can we calibrate agent models to balance tool use and parametric reasoning, mitigating tool overuse while preserving utility?**

4 Method

To address the challenge of tool overuse, we draw inspiration from how humans balance internal knowledge and external tools. Metacognitive theory (Schraw and Moshman, 1995) suggests that human decision-making relies on an implicit awareness of knowledge boundaries, enabling strategic, step-by-step problem-solving (Livingston, 2003).

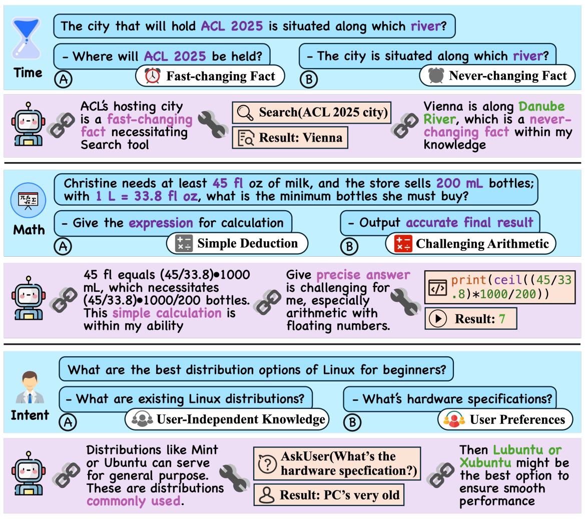

Time
The city that will hold ACL 2025 is situated along which river?
- Where will ACL 2025 be held?
Ⓐ Fast-changing Fact
- The city is situated along which river?
Ⓑ Never-changing Fact
ACL's hosting city is a fast-changing fact necessitating Search tool
Search(ACL 2025 city)
Result: Vienna
Vienna is along Danube River, which is a never-changing fact within my knowledge
Math
Christine needs at least 45 fl oz of milk, and the store sells 200 mL bottles; with 1 L = 33.8 fl oz, what is the minimum bottles she must buy?
- Give the expression for calculation
Ⓐ Simple Deduction
- Output accurate final result
Ⓑ Challenging Arithmetic
45 fl equals (45/33.8)•1000 mL, which necessitates (45/33.8)•1000/200 bottles. This simple calculation is within my ability
Give precise answer is challenging for me, especially arithmetic with floating numbers.
print(ceil((45/33.8)*1000/200))
Result: 7
Intent
What are the best distribution options of Linux for beginners?
- What are existing Linux distributions?
Ⓐ User-Independent Knowledge
- What's hardware specifications?
Ⓑ User Preferences
Distributions like Mint or Ubuntu can serve for general purpose. These are distributions commonly used.
AskUser(What's the hardware specfication?)
Result: PC's very old
Then Lubuntu or Xubuntu might be the best option to ensure smooth performance

Figure 4: Three example queries and their reasoning chains from each domain. The inherent compositionality of a query naturally divides reasoning into knowledge-driven steps and tool-reliant steps.

Inspired by this, we aim to equip agent models with a similar capability—calibrating their metacognition to optimize reasoning and tool use.

To address this, we propose **SMART**, a data-driven approach that enhances self-awareness in agent models. While LLMs acquire broad knowledge from large-scale corpora (Wang et al., 2022), they are not explicitly trained to recognize their own strengths and limitations. To bridge this gap, we introduce **SMART-ER**, the first dataset contrasting areas where models excel versus struggle. Covering three domains with 3K+ questions and structured reasoning chains, SMART-ER helps agents strategically decide when to rely on internal knowledge or external tools.

4.1 Data Collection

To train agents to strategically balance parametric knowledge and external tools within a single reasoning chain, questions must be compositional—blending aspects the model excels at with those it struggles with. Building on prior studies (Hendrycks et al., 2021; Vu et al., 2023; Qian et al., 2024b), we identify three key limitations in LMs: i) *math reasoning*, where models struggle with complex computations requiring precise answers; ii) *temporal knowledge*, as LMs lack access to up-to-date facts beyond their training cutoff; and iii) **user intent understanding**, where implicit preferences cannot be inferred without direct queries. All these challenges necessitate a smarter integration of external tools with the model's rea-

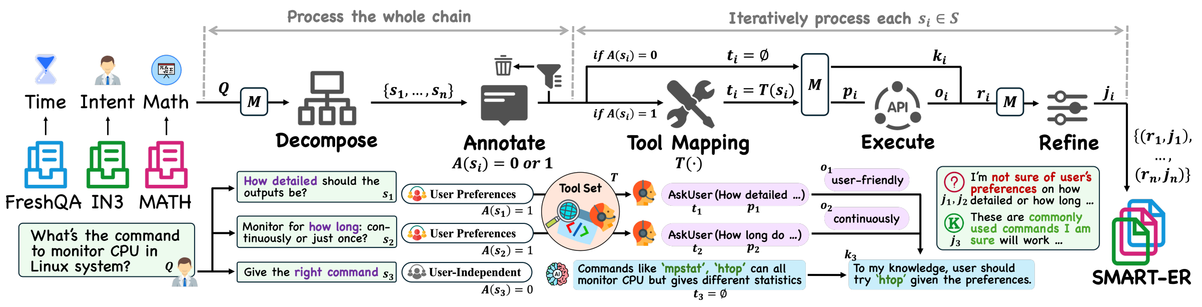

Process the whole chain
Iteratively process each $s_i \in S$
Time
Intent
Math
FreshQA
IN3
MATH
What's the command to monitor CPU in Linux system? Q
Q
M
Decompose
$\{s_1, \dots, s_n\}$
Annotate
$A(s_i) = 0$ or 1
How detailed should the outputs be? s_1
Monitor for how long: continuously or just once? s_2
Give the right command s_3
User Preferences $A(s_1) = 1$
User Preferences $A(s_2) = 1$
User-Independent $A(s_3) = 0$
if $A(s_i) = 0$
if $A(s_i) = 1$
$t_i = \emptyset$
$t_i = T(s_i)$
Tool Mapping
$T(\cdot)$
Tool Set
T
AskUser (How detailed ...) t_1 p_1
AskUser (How long do ...) t_2 p_2
Commands like 'mpstat', 'htop' can all monitor CPU but gives different statistics $t_3 = \emptyset$
M
k_i
p_i
API
Execute
o_i
o_1 user-friendly
o_2 continuously
k_3 To my knowledge, user should try 'htop' given the preferences.
r_i
M
Refine
j_i
j_1, j_2 I'm not sure of user's preferences on how detailed or how long ...
j_3 These are commonly used commands I am sure will work ...
$\{(r_1, j_1), \dots, (r_n, j_n)\}$
SMART-ER

Figure 5: The data pipeline to get SMART-ER. We divide the whole pipeline into several stages for better control and quality of the generated reasoning chain.

Split	Dimension	Math	Time	Intention
Training	*Queries*	2671	1287	997
	T/K Ratio	13/87	20/80	45/55
	Ins. Pair	4717	2574	4398
Test	*Queries*	400	100	100

Table 2: Statistics for SMART-ER. *T/K Ratio* denotes the ratio of tool-reliant to knowledge-driven steps.

soning ability. Building on this insight, we construct data of three domains:

- **Math**: Adapted from MATH (Hendrycks et al., 2021), each query incorporates both challenging math deductions and simple arithmetic to contrast reasoning capabilities.
- **Time**: Adapted from FreshQA (Vu et al., 2023), each query ensures a mix of fast-changing and slow-changing factual knowledge.
- **Intention**: Adapted from Intention-in-Interaction (IN3) (Qian et al., 2024b), each query requires explicit user intent while remaining solvable within the model's capabilities.

This compositional approach helps models calibrate their decision-making by distinguishing when to rely on external tools versus when internal knowledge is sufficient. To illustrate this, we present three example queries from each domain in Figure 4. For details on the question selection and adaptation process, please refer to Appendix B.1.

4.2 Reasoning Chain Construction

As shown in Figure 5, each query Q is decomposed into a structured reasoning plan with n subgoals, $S = \{s_1, s_2, \dots, s_n\}$. This decomposition is enabled by the compositional nature of our queries and is empirically achieved using GPT-4o, an auxiliary model in our pipeline, later denoted as M. Next, for each s_i, we determine whether it requires tool use ($A(s_i) = 1$) or can be resolved with parametric knowledge alone ($A(s_i) = 0$). Using ground truth from existing source data as heuristics, we guide M to annotate each subgoal. During this process, we also discard those queries where all subgoals rely exclusively on either tools or parametric knowledge. After annotating the entire chain, we process each subgoal iteratively, starting from s_1. For each subgoal s_i where $A(s_i) = 1$, we assign an appropriate tool t_i from a predefined tool set using a mapping function $T(\cdot)$:

$$t_i = \begin{cases} T(s_i), & \text{if } A(s_i) = 1 \\ \varnothing, & \text{otherwise} \end{cases}$$

where $t_i = \varnothing$ indicates the model relies solely on its parametric knowledge for reasoning. Empirically, our tool set consists of *Code*, *Search*, and *AskUser*, covering all designed domains.

Next, we proceed with the reasoning process using M. If $A(s_i) = 0$, M reasons over s_i, producing a reasoning step k_i based on its parametric knowledge. Otherwise, we prompt M to generate the necessary parameters p_i for tool invocation, retrieving the tool output o_i. The resulting outcome for each step is formulated as:

$$r_i = \begin{cases} (\, p_i = M(s_i),\ o_i = t_i(p_i) \,), & \text{if } A(s_i) = 1 \\ (\, k_i = M(s_i) \,), & \text{otherwise} \end{cases}$$

where $t_i(\cdot)$ represents the invocation of tool t_i. The iterative process also enables M to incorporate information from prior steps and tool outputs when processing subsequent subgoals, ensuring a coherent and context-aware reasoning flow.

Inspired by metacognitive heuristics that implicitly guide human reasoning, we refine the reasoning chain r_i by explicitly incorporating justifications for whether parametric knowledge suffices or external tool use is necessary. Specifically, we prompt M to generate a justification $j_i = M(s_i, A(s_i))$, conditioned on the subgoal s_i and its annotation

$A(s_i)$. This approach emulates human metacognition by transforming implicit heuristics into explicit natural language explanations, thus enhancing interpretability. Similar to Chain-of-Thought (Wei et al., 2022b) leverages the cumulative probability nature of autoregressive models to guide reasoning, j_i helps the model calibrate its decision-making, improving its ability to strategically balance internal knowledge and external tools.

Finally, by integrating all subgoals, we obtain the complete reasoning chain $R = \{(r_1, j_1), \dots, (r_n, j_n)\}$ for query Q, where each step r_i is either (k_i), indicating a parametric knowledge-driven step, or (p_i, o_i), representing a tool-reliant step. Our method dynamically integrates these steps, ensuring an adaptive balance between internal reasoning and external tool use. To ensure quality, we conduct human supervision on 5% of the data for each step involving M, achieving a pass rate of over 95%. Please refer to Appendix B.2 for details.

4.3 Agent Training Implementation

We partition SMART-ER into training and test splits with statistics in Table 2. For each (Q, R') in the training set, we generate multiple input-output pairs for instruction tuning. The input comprises $\{Q, (r_1, j_1), \dots, (r_{x_i}, j_{x_i})\}$, while the output consists of $\{(r_{x_i+1}, j_{x_i+1}), \dots, (r_{x_{i+1}}, j_{x_{i+1}})\}$, where x_i indexes the tool-reliant steps. This setup ensures iterative reasoning, allowing the agent to leverage prior steps until the next tool invocation or final solution. The number of input-output pairs per (Q, R') also equals the number of tool-reliant steps, facilitating interactive inference.

Using these instruction pairs, we finetune the Llama-3.1 8B and 70B instruct models (Dubey et al., 2024) as well as the Mistral 7B, Nemo(12B) and Small(24B) instruct models (Jiang et al., 2023), adapting them into a family of **SMARTAgent**. These agent models enable interactive tool use, recognizes its own limitations, and balances tool reliance with parametric knowledge-driven reasoning to prevent tool overuse. See Appendix B.3 for training details and hyper-parameters.

5 Experiment

In this section, we present results demonstrating SMARTAgent's effectiveness in reducing tool overuse while enhancing reasoning performance.

5.1 Settings

Data. For in-domain testing, we evaluate SMARTAgent using the test split of adapted SMART-ER data across three domains: Math (MATH), Time (FreshQA), and Intention (IN3). For out-of-distribution (OOD) testing, we assess performance on GSM8K (Cobbe et al., 2021) and MINTQA (He et al., 2024), which test logical reasoning and real-world knowledge.

Baselines. We incorporate three main baselines: i) *Normal Reasoning Trained*: For each domain, we train the model using the training set queries to perform reasoning without tools, leveraging the original solution chain or ground truth. ii) *Base Model Reasoning Prompt*: We directly prompt the model to apply chain-of-thought reasoning without tools to solve the problem. iii) *Base Model Tool Prompt*: We provide the model with all available tools and their usage but allow it to decide independently whether and when to use them.

Inference. For reasoning without tools, the model generates a response including the final answer. For tool-reliant reasoning, the inference is interactive: in each round, if a tool call is detected, we parse and execute it, integrating the tool's output and reasoning into the input. This repeats until the final answer is reached. See Appendix C for details.

Metrics. We use two main evaluation metrics: *Tool Used*, which measures the average number of times a tool is leveraged during reasoning, and *Accuracy*, which evaluates the average performance across queries. For the IN3 dataset, where answers depend on user preferences and lack a single correct response, we adopt the original paper's metrics: *Missing Details Recovery*, assessing whether missing details in vague instructions are recovered, and *Summarized Intention Coverage*, assessing whether the final response covers all user-stated preferences.

5.2 Main Results

We present the main results in Table 3, along with the baseline performance of GPT-4o and GPT-4o-mini for comparison. We also present the OOD results for Mistral-7B and Llama-3.1-8B in Section 5.1, highlighting the following key findings.

SMARTAgent solves tasks efficiently. Compared to the base model in Table 3, which autonomously decides whether to use tools,

Method	**Model**	**Math** (MATH)		**Time** (FreshQA)		**Intention** (Intention-in-Interaction)		
		Tool Used$^{\downarrow}$ (*Times*)	**Accuracy**$^{\uparrow}$ (%)	**Tool Used**$^{\downarrow}$ (*Times*)	**Accuracy**$^{\uparrow}$ (%)	**Tool Used**$^{\downarrow}$ (*Times*)	**Missing Details Recovery**$^{\uparrow}$ **(Lv3 / Lv2, %)**	**Summarized Intention Coverage**$^{\uparrow}$ **(%)**
				Open-Source				
Normal Reasoning Trained	*Mistral-7B*	0.00	17.00	0.00	48.00	0.00	41.86 / 43.84	-
	Llama-3.1-8B	0.00	41.00	0.00	48.00	0.00	38.37 / 42.49	-
Base Model Reasoning Prompt	*Mistral-7B*	0.00	17.25	0.00	29.00	0.00	37.21 / 33.06	-
	Llama-3.1-8B	0.00	53.00	0.00	26.00	0.00	40.70 / 25.76	-
	Mistral-Nemo(12B)	0.00	47.00	0.00	33.00	0.00	44.19 / 28.37	-
	Mistral-Small(24B)	0.00	72.25	0.00	34.00	0.00	41.86 / 31.82	-
	Llama-3.1-70B	0.00	70.00	0.00	36.00	0.00	41.86 / 29.24	-
Base Model Tool Prompt	*Mistral-7B*	3.90	13.25	1.67	49.00	3.80	48.84 / 21.70	63.04
	Llama-3.1-8B	1.93	51.00	2.05	56.00	3.77	54.76 / 25.90	70.20
	Mistral-Nemo(12B)	2.35	46.00	1.19	59.00	1.80	31.35 / 5.82	59.27
	Mistral-Small(24B)	1.55	76.00	1.73	62.00	2.52	45.74 / 33.62	78.20
	Llama-3.1-70B	3.53	67.50	2.08	63.00	2.71	45.74 / 35.96	61.68
SMARTAgent	*Mistral-7B*	0.60$_{\downarrow 3.30}$	22.75$_{\uparrow 5.50}$	1.00$_{\downarrow 0.67}$	64.00$_{\uparrow 15.00}$	3.60$_{\downarrow 0.20}$	74.42$_{\uparrow 25.58}$ / 65.44$_{\uparrow 21.60}$	81.76$_{\uparrow 18.72}$
	Llama-3.1-8B	0.88$_{\downarrow 1.05}$	54.75$_{\uparrow 1.75}$	1.05$_{\downarrow 1.00}$	67.00$_{\uparrow 11.00}$	3.80$_{\uparrow 0.03}$	**81.40**$_{\uparrow 26.64}$ / 67.41$_{\uparrow 24.92}$	78.28$_{\uparrow 8.08}$
	Mistral-Nemo(12B)	0.82$_{\downarrow 1.53}$	49.50$_{\uparrow 2.50}$	1.00$_{\downarrow 0.19}$	**70.00**$_{\uparrow 11.00}$	3.34$_{\uparrow 1.54}$	77.91$_{\uparrow 33.72}$ / 62.15$_{\uparrow 33.78}$	82.30$_{\uparrow 23.03}$
	Mistral-Small(24B)	0.79$_{\downarrow 0.76}$	69.75$_{\downarrow 6.25}$	1.00$_{\downarrow 0.73}$	66.00$_{\uparrow 4.00}$	3.89$_{\uparrow 1.37}$	74.42$_{\uparrow 28.68}$ / **68.87**$_{\uparrow 35.25}$	84.99$_{\uparrow 6.79}$
	Llama-3.1-70B	0.94$_{\downarrow 2.59}$	**72.50**$_{\uparrow 2.50}$	1.01$_{\downarrow 1.07}$	66.00$_{\uparrow 3.00}$	3.51$_{\uparrow 0.80}$	68.60$_{\uparrow 22.86}$ / 58.15$_{\uparrow 22.19}$	**86.09**$_{\uparrow 24.41}$
		Tool Used Macro-Average Decrease (%)		**24.00**		Performance Macro-Average Increase (%)		**37.10**
				Closed-Source				
Base Model Reasoning Prompt	*GPT-4o-mini*	0.00	73.00	0.00	44.00	0.00	45.35 / **32.41**	-
	GPT-4o	0.00	**79.50**	0.00	47.00	0.00	38.37 / 28.54	-
Base Model Tool Prompt	*GPT-4o-mini*	2.55	54.50	1.06	56.00	1.91	**50.00** / 26.90	76.44
	GPT-4o	0.27	79.25	1.01	**65.00**	1.17	40.70 / 15.61	**86.80**

Table 3: SMARTAgent's performance on the test split across three in-domain task categories. The green and red arrows indicate better or worse performance compared to the best baseline method. Its strong performance and fewer tool calls highlight SMARTAgent's efficient and strategic tool use.

Dataset	**GSM8K**		**MINTQA**	
Metrics	**Tool Used**$^{\downarrow}$ (*Times*)	**Accuracy**$^{\uparrow}$ (%)	**Tool Used**$^{\downarrow}$ (*Times*)	**Accuracy**$^{\uparrow}$ (%)
	Llama-3.1-8B			
Normal Reasoning Trained	0.00	80.29	0.00	21.65
Base Model Reasoning Prompt	0.00	82.26	0.00	12.37
Base Model Tool Prompt	2.53	83.17	4.03	16.49
SMARTAgent	**0.76**$_{\downarrow 1.77}$	**83.40**$_{\uparrow 0.23}$	**1.06**$_{\downarrow 2.97}$	**29.90**$_{\uparrow 8.25}$
	Mistral-7B			
Normal Reasoning Trained	0.00	58.68	0.00	21.65
Base Model Reasoning Prompt	0.00	50.57	0.00	19.59
Base Model Tool Prompt	3.56	55.34	6.46	10.31
SMARTAgent	**0.45**$_{\downarrow 3.11}$	**58.98**$_{\uparrow 0.30}$	**0.99**$_{\downarrow 5.47}$	**25.77**$_{\uparrow 4.12}$

Table 4: SMARTAgent's performance on out-of-distribution tasks compared with baseline methods. Results show SMARTAgent can successfully generalize.

SMARTAgent reduces tool usage time per query by 24% on average. At the same time, its performance improves by over 37% across models compared to the best baseline. This demonstrates SMARTAgent's efficiency in tool use, achieving higher results while relying less on external resources.

7B-scale SMARTAgent can outperform GPT-4o baselines. Despite being much smaller, the 7B- and 8B-scale SMARTAgent models can outperform GPT-4o and its 70B counterpart in Time and Intention domains while using fewer tool calls, showcasing their efficient tool use. In Math, where reasoning scales with model size, SMARTAgent lags behind larger models but remains competitive against baselines using the same architecture. These results demonstrate that strategic tool use can bridge the gap between model size and performance, making SMARTAgent a resource-efficient yet powerful alternative.

SMARTAgent generalizes to OOD settings. As shown in Section 5.1, SMARTAgent effectively reduces tool calls while achieving better overall performance on OOD test benchmarks. Notably, SMARTAgent makes only one-fifth the number of tool calls compared to the base model in MINTQA, where tool prompting often leads to excessive reliance and decreased accuracy.

Improper tool uses degrade performance. In the MINTQA and Math domain data, we find that arbitrary tool use can degrade performance compared to standard chain-of-thought reasoning. This aligns with our argument in Section 3 that excessive tool reliance can introduce unpredictable side effects, causing models to struggle with interactive tool calls. As a result, inference may become pro-

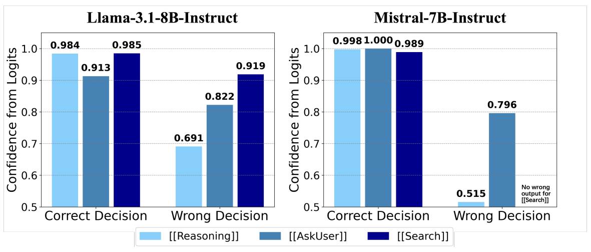


Figure 6: Confidence analysis shows that **SMART** effectively enhances the model's decision-making confidence in selecting the correct reasoning approaches.

longed over multiple rounds, ultimately leading to incorrect answers. Additionally, we observe that larger-scale models, including GPT-4o, use tools less frequently in the Intention domain data, resulting in a greater performance drop than even the 7B-scale SMARTAgent. This may stem from their overconfidence in assisting users, leading them to overlook specific user preferences.

SMARTAgent achieves near-optimal tool use. Datasets such as Time and MINTQA contain up-to-date knowledge necessitating tool use. Ideally, at least one tool call per query is required for a correct answer, and SMARTAgent consistently maintains an average close to one, reflecting near-optimal efficiency. Similarly, in the Intention domain, where queries contain two to four missing details, SMARTAgent invokes tools three times per query, aligning with the expected need.

5.3 Analysis and Case Studies

SMARTAgent effectively reduces tool overuse. Beyond measuring tool use per query, we calculate the tool overuse rate, as defined in Section 3, and report results in Section 5.2 for GSM8K and Math domain test data. Notably, SMARTAgent reduces unnecessary tool calls by up to 50% compared to prompting the base model with tool access. However, despite this reduction, tool overuse persists, which we further examine in error analysis.

Error analysis. We provide error analysis in Table 5, highlighting common failure causes. Tool prompting leads to errors across all categories, while SMARTAgent reduces repetitive calls and improves argument accuracy. However, feedback neglect still causes tool invocation failures, particularly with the *Code* tool, and excessive caution in ensuring calculation accuracy adds overhead. This mirrors human task-solving, where we sometimes rely on calculators despite knowing the steps. Future work may explore balancing convenience, budget, and efficiency to enhance decision-making.

Case Study. In Figure 7, we compare the solution chains of SMARTAgent and the base model with tool prompting. SMARTAgent demonstrates logical planning, context corroboration, and an awareness of its limitations and knowledge boundaries, with clear justifications for its decisions. This metacognitive approach closely mirrors human reasoning processes, making SMARTAgent's reasoning more interpretable and significantly reducing tool use overhead.

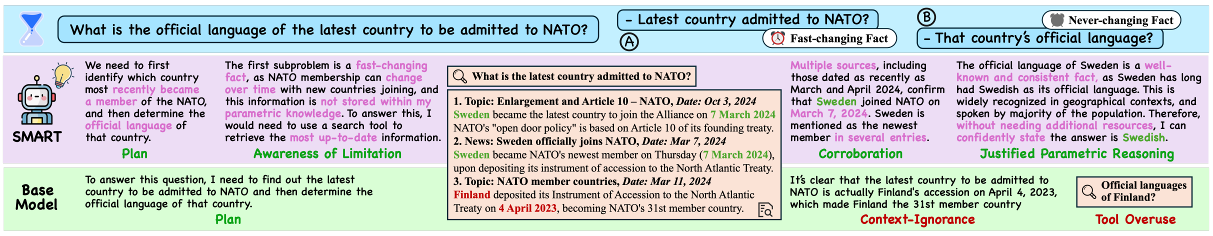
Confidence Validation Experiment. To evaluate SMARTAgents' ability to choose between internal reasoning and tool invocation, we conducted experiments using special tokens to analyze decision confidence. Specifically, we trained the model on Time and Intention domains, introducing special tokens: "[[Reasoning]]" for internal reasoning, "[[AskUser]]" for the *AskUser* tool, and "[[Search]]" for the *Search* tool. These tokens, prepended at each step, guided decision-making during training (see Appendix C.5). For evaluation, we sampled 50 decision steps from both domains' test splits, measuring confidence via token logits. Decisions were categorized as correct or incorrect based on alignment with ground truth. As shown in Figure 6, the model exhibited higher confidence in correct decisions, demonstrating **SMART's effectiveness in boosting confidence and distinguishing between internal knowledge and tool use**.

6 Discussions

Agent's improper tool usage. Our empirical analysis reveals a notable phenomenon of *tool overuse*, where agents frequently rely on external tools even when internal knowledge is sufficient. This over-reliance likely arises from two factors: i) the agent's uncertainty about its own capabilities, and ii) the perceived ease of external lookups compared to internal reasoning. We also observe instances of *tool underuse*, especially in large-scale models like GPT-4o and Llama-70B, where agents neglect to call essential tools, possibly due to misjudging the complexity of the task. Both overuse and underuse contribute to concerns over computational efficiency and solution accuracy. Future research could explore methods to better balance these trade-offs, such as by introducing explicit resource constraints or budgets for tool calls.

Error Type (Explanation)	**Case / Model Action**	**Wrong Reason**	**Common Seen**
Repetitive Tool Calls Uses the same query to call the tool for multiple times.	**Last Call:** Search(current richest person) **Reasoning:** several people are mentioned instead of one richest, search again… **Tool Call:** Search(current richest person)	The model fails to extract the most useful information and instead relies on repetitive calls.	*Domain*: Time Tool Prompt
Ignorance of Feedback Overlooks tool feedback and fails to correct erroneous behavior.	**Last Output:** Error! Traceback: function 'ceil' not found **Tool Call:** Code(```print(ceil(45/33.8•5))```)	The error persists due to the absence of 'from math import ceil,' causing an incorrect call.	*Domain*: Math Tool Prompt, SMARTAgent
Tool Calls on Simple Subgoal Invokes tool calls for subgoals that are considered trivial by the user.	**Reasoning:** I need to use code to ensure the accuracy of my calculation. **Tool Call:** Code(```print(30•40/2)```)	Still using tool calls on simple calculation to ensure accuracy.	*Domain*: Math Tool Prompt, SMARTAgent
Inaccurate Tool Call Arguments Employs imprecise arguments that causes deviations in the solution chain.	**Query:** Find the next music festival happening in my city. **Tool Call:** AskUser(what's your favorite music)	Ask about not-related trivial details instead of where the city is, date or time frame, etc.	*Domain*: Intention Tool Prompt

Table 5: Error analysis of common task failure causes, with explanations and examples.


What is the official language of the latest country to be admitted to NATO?

- Latest country admitted to NATO? Ⓐ Fast-changing Fact

Ⓑ Never-changing Fact - That country's official language?

SMART

We need to first identify which country most recently became a member of the NATO, and then determine the official language of that country.
Plan

The first subproblem is a fast-changing fact, as NATO membership can change over time with new countries joining, and this information is not stored within my parametric knowledge. To answer this, I would need to use a search tool to retrieve the most up-to-date information.
Awareness of Limitation

What is the latest country admitted to NATO?

1. Topic: Enlargement and Article 10 – NATO, *Date: Oct 3, 2024*
Sweden became the latest country to join the Alliance on 7 March 2024 NATO's "open door policy" is based on Article 10 of its founding treaty.
2. News: Sweden officially joins NATO, *Date: Mar 7, 2024*
Sweden became NATO's newest member on Thursday (7 March 2024), upon depositing its instrument of accession to the North Atlantic Treaty.
3. Topic: NATO member countries, *Date: Mar 11, 2024*
Finland deposited its Instrument of Accession to the North Atlantic Treaty on 4 April 2023, becoming NATO's 31st member country.

Multiple sources, including those dated as recently as March and April 2024, confirm that Sweden joined NATO on March 7, 2024. Sweden is mentioned as the newest member in several entries.
Corroboration

The official language of Sweden is a well-known and consistent fact, as Sweden has long had Swedish as its official language. This is widely recognized in geographical contexts, and spoken by majority of the population. Therefore, without needing additional resources, I can confidently state the answer is Swedish.
Justified Parametric Reasoning

Base Model

To answer this question, I need to find out the latest country to be admitted to NATO and then determine the official language of that country.
Plan

It's clear that the latest country to be admitted to NATO is actually Finland's accession on April 4, 2023, which made Finland the 31st member country
Context-Ignorance

Official languages of Finland?
Tool Overuse

Figure 7: Case study comparing the performance of Tool Prompting and SMARTAgent.

Dataset	**Model**	**Tool Abuse**↓	
		Tool Prompt	**SMARTAgent**
GSM8K	*Llama-3.1-8B*	66.20	**37.77**
	Mistral-7B	30.25	**15.76**
Math (MATH)	*Llama-3.1-8B*	35.59	**35.22**
	Mistral-7B	12.47	**7.77**

Table 6: Statistics on tool overuse, defined in Section 3.

Mechanisms behind human and LM's decision-making. Cognitive science suggests that human decision-making arises from both intuitive judgments and reflective strategies. Similarly, in language models (LMs), problem-solving is influenced by implicit heuristics (e.g., memorized patterns) and explicit tool-using behaviors. When tools are available, LMs often default to external queries, akin to humans seeking external confirmation when uncertain. However, unlike humans, LMs lack self-monitoring and rely on external or data-driven cues to determine when to trust their internal knowledge. Developing frameworks that integrate implicit heuristics with explicit reasoning could lead to more adaptive and efficient decision-making in LMs.

Enhancement of model's self-awareness. Our data-driven calibration strategy, which provides explicit rationales for when to rely on internal knowledge versus external tools, shows promising results. Other approaches, such as confidence probing via logits, integration of specialized self-checking modules, or reinforcement learning from feedback, might also refine tool usage thresholds. Future research could investigate how these signals affect the model's internal distributions and identify representations that capture the *awareness* of boundaries. Additionally, iterative or in-context learning could allow real-time metacognitive calibration, offering a more efficient safeguard against both overuse and underuse of resources.

7 Conclusion

Inspired by human metacognition in decision-making, we propose the SMART paradigm for agent reasoning, where agents recognize their knowledge boundaries to decide when to use tools or parametric knowledge. Specifically, SMART-ER refines this decision boundary by incorporating questions that highlight areas where current LMs excel and struggle. Using these curated reasoning chains, we train SMARTAgent to better balance tool use and parametric knowledge, reducing tool overuse. Our results show that a simple data-driven approach can effectively calibrate model awareness, paving the way for efficient, low-resource agent development where "smartness" stems from both performance and metacognitive ability to optimize the reasoning strategy.

Limitations

Our study focuses on three key domains where LLMs explicitly struggle—Math, Intention, and Time—building on insights from existing literature. However, LLMs also face challenges in areas such as long-tail knowledge and domain-specific expertise, where external resources are essential. Expanding SMART-ER to these domains could further refine model self-awareness and improve calibration in knowledge boundary, complementing the strong OOD performance that SMARTAgent has already demonstrated. Additionally, while we evaluate our approach on two major model families, extending our analysis to a broader range of architectures, including Qwen, DeepSeek, and varying model sizes, could further validate and enhance the generalizability of our findings.

Acknowledgment

This research is based upon work supported DARPA ITM Program No. FA8650-23-C-7316, DARPA ECOLE Program No. #HR00112390060 and the Molecule Maker Lab Institute: an AI research institute program supported by NSF under award No. 2019897. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Alfonso Amayuelas, Kyle Wong, Liangming Pan, Wenhu Chen, and William Wang. 2023. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models. *arXiv preprint arXiv:2305.13712*.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2024. Large language models as tool makers. In *The Twelfth International Conference on Learning Representations*.

Lida Chen, Zujie Liang, Xintao Wang, Jiaqing Liang, Yanghua Xiao, Feng Wei, Jinglei Chen, Zhenghong Hao, Bing Han, and Wei Wang. 2024a. Teaching large language models to express knowledge boundary from their own signals. *arXiv preprint arXiv:2406.10881*.

Xiusi Chen, Jyun-Yu Jiang, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, and Wei Wang. 2024b. Min-Prompt: Graph-based minimal prompt data augmentation for few-shot question answering. pages 254–266, Bangkok, Thailand.

Xiusi Chen, Yu Zhang, Jinliang Deng, Jyun-Yu Jiang, and Wei Wang. 2023. Gotta: generative few-shot question answering by prompt-based cloze data augmentation. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pages 909–917. SIAM.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Jie He, Nan Hu, Wanqiu Long, Jiaoyan Chen, and Jeff Z. Pan. 2024. MINTQA: A multi-hop question answering benchmark for evaluating llms on new and tail knowledge. *arXiv preprint arXiv:2412.17032*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, et al. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Moxin Li, Yong Zhao, Yang Deng, Wenxuan Zhang, Shuaiyi Li, Wenya Xie, See-Kiong Ng, and Tat-Seng Chua. 2024. Knowledge boundary of large language models: A survey. *arXiv preprint arXiv:2412.12472*.

Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, et al. 2023. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *arXiv preprint arXiv:2305.18703*.

Jennifer Livingston. 2003. Metacognition: An overview.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2022. A survey of deep learning for mathematical reasoning. *arXiv preprint arXiv:2212.10535*.

Marvin Minsky. 1986. *The Society of Mind*. Simon & Schuster.

Kangyun Ning, Yisong Su, Xueqiang Lv, Yuanzhe Zhang, Jian Liu, Kang Liu, and Jinan Xu. 2024. Wtu-eval: a whether-or-not tool usage evaluation benchmark for large language models. *arXiv preprint arXiv:2407.12823*.

Mathematical Association of America (MAA). 2023. American mathematics competitions.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. 2024. Training software engineering agents and verifiers with swe-gym. In *arxiv*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025a. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Cheng Qian, Hongyi Du, Hongru Wang, Xiusi Chen, Yuji Zhang, Avirup Sil, Chengxiang Zhai, Kathleen McKeown, and Heng Ji. 2025b. Modelingagent: Bridging llms and mathematical modeling for real-world challenges. *arXiv preprint arXiv:2505.15068*.

Cheng Qian, Chi Han, Yi Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. 2023a. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6922–6939.

Cheng Qian, Peixuan Han, Qinyu Luo, Bingxiang He, Xiusi Chen, Yuji Zhang, Hongyi Du, Jiarui Yao, Xiaocheng Yang, Denghui Zhang, et al. 2024a. Escapebench: Pushing language models to think outside the box. *arXiv preprint arXiv:2412.13549*.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, et al. 2024b. Tell me more! towards implicit user intention understanding of language model driven agents. *arXiv preprint arXiv:2402.09205*.

Cheng Qian, Shihao Liang, Yujia Qin, Yining Ye, Xin Cong, Yankai Lin, Yesai Wu, Zhiyuan Liu, and Maosong Sun. 2024c. Investigate-consolidate-exploit: A general strategy for inter-task agent self-evolution. *arXiv preprint arXiv:2401.13996*.

Cheng Qian, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2024d. Toolink: Linking toolkit creation and using through chain-of-solving on open-source model. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 831–854.

Cheng Qian, Xinran Zhao, and Sherry Tongshuang Wu. 2023b. "merge conflicts!" exploring the impacts of external distractors to parametric knowledge graphs. *arXiv preprint arXiv:2309.08594*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. 2023. Tool learning with foundation models. *arXiv preprint arXiv.2304.08354*, 10.

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

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

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. Investigating the factual knowledge boundary of large language models with retrieval augmentation. *arXiv preprint arXiv:2307.11019*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Gregory Schraw and David Moshman. 1995. Metacognitive theories. *Educational psychology review*, 7:351–371.

Yuanhao Shen, Xiaodan Zhu, and Lei Chen. 2024. Smartcal: An approach to self-aware tool-use evaluation and calibration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 774–789.

AgentGPT Team. 2024. Agentgpt.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

XAgent Team. 2023. Xagent: An autonomous agent for complex task solving. *XAgent blog*.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, et al. 2023. Freshllms: Refreshing large language models with search engine augmentation. *arXiv preprint arXiv:2310.03214*.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. 2025a. Otc: Optimal tool calls via reinforcement learning. *arXiv preprint arXiv:2504.14870*.

Hongru Wang, Rui Wang, Boyang Xue, Heming Xia, Jingtao Cao, Zeming Liu, Jeff Pan, and Kam-Fai Wong. 2024a. Appbench: Planning of multiple apis from various apps for complex user instruction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15322–15336.

Hongru Wang, Boyang Xue, Baohang Zhou, Rui Wang, Fei Mi, Weichao Wang, Yasheng Wang, and Kam-Fai Wong. 2024b. UniRetriever: Multi-task candidates selection for various context-adaptive conversational retrieval. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 17074–17086, Torino, Italia. ELRA and ICCL.

Hongru Wang, Boyang Xue, Baohang Zhou, Tianhua Zhang, Cunxiang Wang, Huimin Wang, Guanhua Chen, and Kam-Fai Wong. 2024c. Self-DC: When to reason and when to act? self divide-and-conquer for compositional unknown questions. *arXiv preprint arXiv:2402.13514*.

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and S Yu Philip. 2022. Generalizing to unseen domains: A survey on domain generalization. *IEEE transactions on knowledge and data engineering*, 35(8):8052–8072.

Xingyao Wang, Boxuan Li, Yufan Song, Xiangru Tang, Frank F. Xu, Bowen Li, Jiayi Pan, Mingchen Zhuge, Niklas Muennighoff, Yizhe Zhang, Ren Ma, Hoang H. Tran, Yanjun Shao, Bill Qian, Fuqiang Li, Jaskirat Singh, Yueqi Song, Mingzhang Zheng, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. 2025b. Openhands: An open platform for ai software developers as generalist agents. In *Proc. The Thirteenth International Conference on Learning Representations (ICLR2025)*.

Zekun Wang, Ge Zhang, Kexin Yang, Ning Shi, Wangchunshu Zhou, Shaochun Hao, Guangzheng Xiong, Yizhi Li, Mong Yuan Sim, Xiuying Chen, et al. 2023. Interactive natural language processing. *arXiv preprint arXiv:2305.13246*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022a. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

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

Shujin Wu, May Fung, Cheng Qian, Jeonghwan Kim, Dilek Hakkani-Tur, and Heng Ji. 2025. Aligning llms with individual preferences via interaction. In *Proc. The 31st International Conference on Computational Linguistics (COLING2025)*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Boyang Xue, Fei Mi, Qi Zhu, Hongru Wang, Rui Wang, Sheng Wang, Erxin Yu, Xuming Hu, and Kam-Fai Wong. 2024. Ualign: Leveraging uncertainty estimations for factuality alignment on large language models. *arXiv preprint arXiv:2412.11803*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

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

Xunjian Yin, Xu Zhang, Jie Ruan, and Xiaojun Wan. 2024. Benchmarking knowledge boundary for large language model: A different perspective on model evaluation. *arXiv preprint arXiv:2402.11493*.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuan-Jing Huang. 2023. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665.

Fei Yu, Hongbo Zhang, Prayag Tiwari, and Benyou Wang. 2024. Natural language reasoning, a survey. *ACM Computing Surveys*, 56(12):1–39.

Pengfei Yu and Heng Ji. 2024. Information association for language model updating by mitigating LM-logical discrepancy. In *Proceedings of the 28th Conference on Computational Natural Language Learning*, pages 117–129, Miami, FL, USA. Association for Computational Linguistics.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R. Fung, Hao Peng, and Heng Ji. 2024. Craft: Customizing llms by creating and retrieving from specialized toolsets. In *Proc. The Twelfth International Conference on Learning Representations (ICLR2024)*.

Hanning Zhang, Shizhe Diao, Yong Lin, Yi Fung, Qing Lian, Xingyao Wang, Yangyi Chen, Heng Ji, and Tong Zhang. 2024. R-tuning: Instructing large language models to say 'i don't know'. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7106–7132.

Appendix

A Preliminary Study Details

A.1 Agent Experiment Details

The system instruction that we provide to both the XAgent and AgentGPT is:

Prompt for Agent Preliminary Study

Solve the following task accurately, and use tools to help you only if necessary.

For LM-driven agent systems, we first prompt GPT-4o with all the questions from the GSM8K test set without using any tools. We then filter out only the questions that GPT-4o can correctly answer through pure text-based reasoning. From this refined dataset, we randomly sample 50 questions to evaluate AgentGPT and XAgent's performance. Surprisingly, despite the core model being capable of solving all sampled questions without external tools, it still heavily relies on tools during reasoning, leading to tool overuse.

A.2 Model Experiment Details

For both Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3, we prompt the model to do inference two times for each question from GSM8K's test set. The first time we instruct the model to reason normally to solve the query with the following system instruction:

Prompt for Model Preliminary Study (Normal)

You are an advanced assistant designed to solve tasks autonomously using your knowledge and reasoning. Clearly articulate your thought process and reasoning steps before presenting the final response to ensure transparency and accuracy.

The second time, we give the model access to tools and instruct it to independently decide when to use them based on the following system instruction:

Prompt for Model Preliminary Study (Tool)

### Task
You are a highly capable assistant designed to solve tasks effectively using your knowledge and available tools.

### Principles
1. Reason Independently:
• Leverage your own knowledge to analyze and solve reasoning steps whenever possible. Use external tools only when necessary.
2. Tool Usage:
• Use code snippet "'python ... "' to write, execute a python code snippet, and retrieve the result from its printed output.
3. Step-by-Step Approach:
• Work through reasoning systematically, breaking down the task into manageable steps. Rely on your knowledge until a gap is identified that requires tool support. Employ tools to address gaps and integrate the findings into your solution.
4. Goal-Oriented Resolution:
• Conclude your reasoning process by achieving a clear, accurate, and succinct solution based on your independent analysis and insights gained from tools.

### Output Guidelines
• If you need to use the code tool, please wrapped it "'python ... "' and write the code snippet inside. Make sure you include all the packages necessary and the code is executable. And then you should stop generating.
• If you just begin to generate reasoning steps, please directly reason after "### Reasoning Steps".
• If you are generating after the output of a code snippet, please continue to do the reasoning in your output, you can still call the tool if necessary.
• Finally you should give a succinct and accurate final response to directly address the task after "### Final Response".

We provide a code-writing and execution environment, specifically designed to assist with complex math tasks and calculations. Whenever the model generates a code snippet in its output, we parse and execute it, returning the result. The model then continues reasoning based on its previous steps and the executed output. This process iterates until a final response is reached.

B Data Construction Details

B.1 Data Selection

For the **Math** domain, we first collect questions that the current GPT model answers incorrectly, ensuring their inherent difficulty. We then decompose the ground truth reasoning chain to assess the complexity of each step, selecting questions that contain both straightforward and challenging aspects to provide a balanced reasoning task.

For the **Time** domain, we filter out all questions explicitly labeled as involving fast-changing facts. Given the limited number of such questions, we further augment the dataset using a self-instruct approach, prompting the GPT model to generate additional queries related to rapidly evolving information. To introduce compositional reasoning, each generated query is expanded with an additional sub-question involving well-established, slow-changing facts, forming multi-hop queries that require a nuanced understanding of temporal knowledge.

For the **Intention** domain, we filter out all queries labeled as vague in task definition, particularly those requiring explicit user clarification. To ensure that each query remains solvable without tool reliance, we probe GPT to verify that the model can generally answer each selected question without application of tools. This filtering process refines the dataset to only include queries where the model's performance is not hindered by a lack of inherent capability but rather by the absence of user-provided intent.

The data adaptation process is fully automated, with manual checks conducted on 5% of the samples at each stage to ensure the quality of the final filtered questions.

B.2 Reasoning Chain Construction

Empirically, we incorporate three tools in our constructed tool set:

- **Code**: An environment for code writing and execution, enhancing the model's capability in complex calculations, equation solving, and related tasks. To use this tool, the model must generate an executable code snippet within `”'python <code> ”'` and print the output to obtain the execution results.
- **Search**: A real-time web search tool for retrieving the most up-to-date factual knowledge or information beyond the model's parametric knowledge. To invoke this tool, the model should provide a search query in the format `Search(<query>)` to obtain relevant search engine results. We empirically use the Serper API as the backend search engine.
- **AskUser**: A tool for querying the user to clarify intentions, preferences, or general inquiries. This tool enables the model to retrieve user-provided responses by issuing a user-oriented query in the format `AskUser(<query>)`. To simulate user responses in our experiments, we employ a GPT model as the backend.

From the constructed reasoning chains, we empirically observe that the **Code** tool is mainly used in the **Math** domain, the **Search** tool is mainly utilized in the **Time** and **Intention** domains, while the **AskUser** tool is mainly employed in the **Intention** domain.

For each step involving the auxiliary model M, we manually verify data quality to ensure: i) tasks are decomposed into fine-grained, reasonable sub-goals, ii) tool-calling formats are correct, and iii) justifications align with labels and accurately explain why parametric knowledge suffices or a specific tool is required. Through iterative optimization of instructions to M, we achieve a final pass rate exceeding 95%.

Hyperparameter	**Value**
Models	Llama-3.1-8B, Mistral-7B
Fine-tuning Method	SFT
PEFT	LoRA
LoRA Rank	16
LoRA Alpha	32
LoRA Dropout	0.05
LoRA Target	All Layers
Sequence Length (cutoff_len)	4096 tokens
Batch Size (Per Device)	2
Gradient Accumulation Steps	4
Learning Rate	1e-4
Learning Rate Scheduler	Cosine
Warmup Ratio	0.1
Number of Epochs	3
Precision	bfloat16

Table 7: Hyperparameters during Fine-Tuning.

B.3 Training

For fine-tuning, we used **Llama-3.1-8B-Instruct**[2], **Llama-3.1-70B-Instruct**, **Mistral-7B-Instruct-v0.3**[3], **Mistral-Nemo-Instruct-2407**[4], and **Mistral-Small-24B-Instruct-2501**[5] as base models. We applied supervised fine-tuning (SFT) in the Alpaca instruction-following format (Instruction-Input-Output), computing the loss only on tokens in the Output field.

Training was conducted on 4 NVIDIA A40 GPUs using LoRA (Low-Rank Adaptation) with a rank of 16 and an alpha of 32, applied across all model layers. The maximum sequence length

[2] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

[3] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3

[4] https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407

[5] https://huggingface.co/mistralai/Mistral-Small-24B-Instruct-2501

was set to 4096 tokens, and models were trained for 3 epochs with a learning rate of 1e-4, using a cosine learning rate scheduler with a 10% warmup ratio. To manage memory constraints, we set a per-device batch size of 2 and applied gradient accumulation over 4 steps. Training used `bfloat16` (bf16) precision, with evaluations every 100 steps, using 1% of the dataset for validation. Fine-tuning hyperparameters are detailed in Table 7.

The system instruction for finetuning is presented in the following:

System Instruction for Training

You are a highly capable assistant designed to solve tasks effectively using your knowledge and available tools. Follow these principles:

1. Reason Independently: Leverage your own knowledge to analyze and solve reasoning steps whenever possible. Use external tools only when necessary.
2. Tool Usage:
<Specific Tool Description>
3. Step-by-Step Approach:
• Work through reasoning systematically, breaking down the task into manageable steps.
• Rely on your knowledge until a gap is identified that requires tool support.
• Employ tools to address gaps and integrate the findings into your solution.
4. Goal-Oriented Resolution:
Conclude your reasoning process by achieving a clear, accurate solution based on your independent analysis and insights gained from tools. After your reasoning, provide your response to directly address the task.

Your reasoning should be transparent, logical, and concise. Stop and document the reasoning whenever you need to use a tool to gather more information. Continue until you reach the final solution and give final response.

C Experiment Details

C.1 Data Setting

For in-domain testing, we use a subset of adapted SMART-ER data. Specifically, for the Math domain, we randomly sample 400 test instances from MATH, ensuring coverage of all testing categories (algebra, geometry, number theory, etc.), while spanning five difficulty levels. For the Time domain, we select 100 randomly sampled adapted data points from FreshQA, ensuring that each instance incorporates both fast-changing and slow-changing aspects. For the Intention domain, we randomly sample 100 data points from Intention-in-Interaction, ensuring that all selected instructions are vague and require specific user preferences to resolve.

For out-of-domain testing, we directly use the full test set of GSM8K without modifications. For MINTQA, due to its large size, we randomly sample 10% of the data points that meet the following criteria: the question requires multi-hop reasoning and contains both old and new knowledge. This selection ensures a challenging test set that evaluates the model's ability to generalize beyond in-domain tasks while maintaining a focus on complex reasoning and real-world knowledge retrieval.

C.2 Baselines

For the baseline *Normal Reasoning Trained*, we train a separate model for each domain. Specifically, for Math, Time, and Intention, we use the same queries as in the SMART-ER training set. In the Math domain, we leverage existing solution chains from the MATH dataset as training data. For the IN3 and Time domains, we use GPT-4o to generate normal reasoning chains, guided by existing annotations on final answers or missing details as heuristics. These domain-specific solution chains are then used to train the model.

For the baseline *Base Model Reasoning Prompt*, we use the following system instruction to evaluate the model's performance:

Base Model Reasoning Prompt

You are an advanced assistant designed to solve tasks autonomously using your knowledge and reasoning. Clearly articulate your thought process and reasoning steps before presenting the final response to ensure transparency and accuracy.
In the field '### Reasoning Steps', clearly articulate your thought process and reasoning steps towards the final answer. Then you should present a succinct and accurate final response in the field '### Final Response'.

For the baseline *Base Model Tool Prompt*, we use the same system prompt as in appendix A.2, allowing the model to access tools and freely decide whether and when to use them.

C.3 Interactive Inference

For both the baseline *Base Model Tool Use* and our *SMARTAgent*, we adopt an interactive approach for inference. Specifically, we first prompt the target model with the query and obtain its output. In this output, we use a rule-based natural language matching method to determine whether a tool call or a final answer is present (e.g., detecting whether "### Final Response" appears in the output to identify the final response).

If the final response is found, we extract it and terminate the iterative process. If a tool call is detected, we parse the parameters provided by the model to execute the tool call. Based on the specific tool's name, we invoke the corresponding API and integrate its output into the model's response. Next, we append the model's reasoning before the tool call, the tool call itself, and its output to the model's input. We then re-prompt the model to continue reasoning, given the previously executed tool call and its result.

This iterative process continues until the final response is successfully parsed and retrieved, forming the complete interactive inference process.

Below, we illustrate the respective input and output in an iterative inference process consisting of two iterations:

Interactive Inference

```
——— Iterate 2 Input Begin ———

— Iterate 1 Input Begin —
### Task
<target task>
### Reasoning Steps
— Iterate 1 Input End —

== Iterate 1 Output Begin ==
- Step 1: <title>, general reasoning
<reasoning>
- Step 2: <title>, tool: <tool name>
<tool call parameter>
== Iterate 1 Output End ==
- Output: <tool execution output>

——— Iterate 2 Input End ———

====== Iterate 2 Output Begin ======
- Step 3: <title>, general reasoning
<reasoning>
- Step 4: ...
...
...
### Final Response
<final answer>
====== Iterate 2 Output End ======
```

C.4 Additional Results

We also provide results from the latest **Llama-3.3-70B-Instruct**[6] model in Appendix C.3, comparing its performance with the **Llama-3.1-70B-Instruct**-based SMARTAgent. Although **Llama-3.3** is the newest version, we use the **3.1** series to maintain consistency with the **8B** model, which is also from the **3.1** version. Empirically, we found no significant difference in performance between the **3.3** and **3.1** versions of the **70B** model.

[6]https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct

C.5 Confidence Validation

We independently train the Llama-3.1-8B-Instruct and Mistral-7B-Instruct models with the added special tokens. At each reasoning step, we prepend a special token at the very beginning to indicate the model's chosen approach—whether it relies on external tools (e.g., "[[AskUser]]" or "[[Search]]") or its own parametric knowledge (e.g., "[[Reasoning]]").

By analyzing the probability of generating each special token, we can assess the model's confidence in its decision-making process. Apart from the added special tokens, the rest of the original reasoning chain remains unchanged, maintaining the following structured format:

Step Format

```
- Step <index>: [[Special Token]] <title>
<content>
- Step <index>: ...
```

We train the model using the exact same hyperparameter setting introduced in Appendix B.3. During inference, we randomly sample 50 decision-making steps from the test split of both the *Time* and *Intention* domains. A decision-making step refers to the final action in a reasoning sequence—given the previous $n-1$ steps, we evaluate whether the model correctly decides between using a tool or relying on its parametric knowledge for the nth step. This evaluation is performed within the context of the full solution chain, which consists of m steps in total ($m \geq n$).

D Additional Evaluation and Analysis

To address concerns regarding dataset and model selection, we conducted additional experiments targeting two key areas: (1) the applicability of SMARTAgent on more complex reasoning benchmarks beyond GSM8K, and (2) the behavior of o1-like models with respect to tool use.

D.1 Evaluation on Advanced Reasoning Dataset

To assess SMARTAgent's performance on more challenging reasoning tasks, we evaluated it on the AMC'23 benchmark (of America , MAA), a dataset known for its mathematical complexity and nuanced problem-solving requirements.

Method	Model	Math (MATH)		Time (FreshQA)		Intention (Intention-in-Interaction)		
		Tool Used$^{\downarrow}$ (*Times*)	Accuracy$^{\uparrow}$ (%)	Tool Used$^{\downarrow}$ (*Times*)	Accuracy$^{\uparrow}$ (%)	Tool Used$^{\downarrow}$ (*Times*)	Missing Details Recovery$^{\uparrow}$ (Lv3 / Lv2, %)	Summarized Intention Coverage$^{\uparrow}$ (%)
Open-Source								
SMARTAgent	*Llama-3.1-70B*	0.94	72.50	1.01	**66.00**	3.51	**68.60** / 58.15	**86.09**
	Llama-3.3-70B	0.61	**76.25**	1.00	65.00	3.15	61.63 / **59.01**	84.45

Table 8: Performance of SMARTAgent when using Llama-3.3-70B-Instruct as the base model, compared to the original results with its Llama-3.1-70B-Instruct counterpart.

We tested two base models: Llama-3.1-8B-Instruct and Mistral-Nemo-Instruct. We compared SMARTAgent against a baseline tool prompting strategy where the tool is always made available without dynamic control.

Method	Llama-3.1-8B-Instruct	Mistral-Nemo-Instruct
Tool Prompt Baseline	12.50	15.00
SMARTAgent	**17.50**	**20.00**

Table 9: Accuracy (%) on AMC (2023) benchmark.

Table 9 shows that SMARTAgent outperforms the baseline across both model backbones, highlighting its ability to handle complex tasks with improved reasoning-tool use balance.

D.2 Behavior of o1-like Models

To further investigate different model's tool use behavior, we conducted experiments on Deepseek-R1-Distilled variants of Llama and Qwen. Surprisingly, these models exhibited a tendency toward *tool underuse*, contrary to the overuse issue our paradigm primarily targets.

Model	Time	AMC (2023)
Deepseek-R1-Distilled-Llama	60.00	12.50
Deepseek-R1-Distilled-Qwen	72.00	37.50

Table 10: Tool Underuse Rate (%) on Time and AMC tasks.

We define tool underuse as the rate at which a model fails to invoke a tool in scenarios where tool use would be expected. As shown in Table 10, both distilled models significantly underutilize tools, which we attribute to potential overfitting to parametric reasoning, a phenomenon aligned with "overthinking" reported in prior literatures.

D.3 SMARTAgent Adaptation for Distilled Models

To further test the adaptability of SMARTAgent, we fine-tuned these distilled models using our SMART paradigm and evaluated them on a time-domain QA benchmark.

Method	Distilled-Llama	Distilled-Qwen
Base Model Reasoning Prompt	30.00	12.00
Base Model Tool Prompt	36.00	26.00
SMARTAgent	**40.00**	**52.00**

Table 11: Accuracy (%) on Time-domain QA with Distilled Models.

Table 11 shows that SMARTAgent enhances both models' performance, demonstrating its effectiveness not only in mitigating overuse but also in addressing underuse by promoting strategic tool engagement. These findings reinforce SMART's broader applicability across diverse reasoning paradigms and model types.