Alignment for Efficient Tool Calling of Large Language Models

Anonymous ACL submission

Abstract

Recent advancements in tool learning have enabled large language models (LLMs) to integrate external tools, enhancing their task performance by expanding their knowledge boundaries. However, relying on tools often introduces trade-offs between performance, speed, and cost, with LLMs sometimes exhibiting overreliance and overconfidence in tool usage. This paper addresses the challenge of aligning LLMs with their knowledge boundaries to make more intelligent decisions about tool invocation. We propose a multi-objective alignment framework that combines probabilistic knowledge boundary estimation with dynamic decision-making, allowing LLMs to better assess when to invoke tools based on their confidence. Our framework includes two methods for knowledge boundary estimation—consistency-based and absolute estimation—and two training strategies for integrating these estimates into the model's decision-making process. Experimental results on various tool invocation scenarios demonstrate the effectiveness of our framework, showing significant improvements in tool efficiency by reducing unnecessary tool usage.

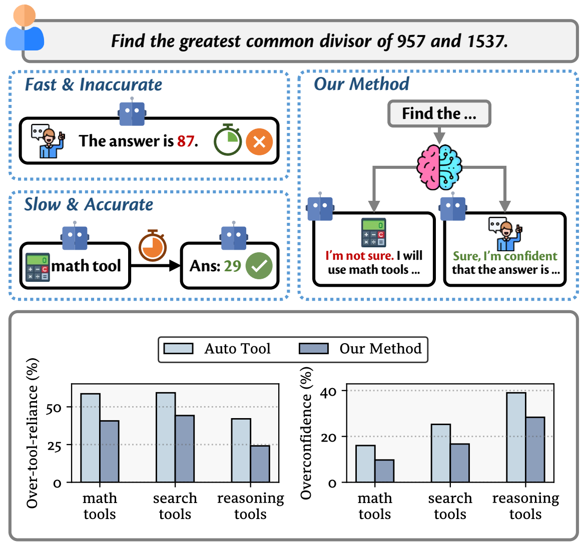

Figure 1: Our method effectively enables LLMs to switch between answering independently and calling tools (upper part), thereby reducing the model's over-reliance and overconfidence in tools (lower part).

1 Introduction

The objective of tool learning is to enable large language models (LLMs; Gemini Team, 2023; Achiam et al., 2023; Dubey et al., 2024) to acquire the capability to effectively utilize external tools, thereby enhancing their performance across various downstream tasks (Schick et al., 2023; Hao et al., 2023; Hsieh et al., 2023; Tang et al., 2023). Tools can be regarded as extensions of an LLM's knowledge or capability boundaries. By invoking tools, models can accomplish tasks beyond their knowledge boundaries and even access information from different modalities (Zeng et al., 2022).

While tools can enhance LLM's task performance, it is important to note that solving tasks through tool invocation often requires more steps, longer completion times, and additional tool-calling costs. For example, in question-answering scenarios involving search tools, the model must first generate a query for the retrieval tool, wait for the search results, and then process these results to produce a final answer. In contrast, direct answering involves simply generating a response. This introduces a trade-off problem between performance and speed. Unfortunately, recent studies have shown that O1-like LLMs struggle to strike a balance between these two aspects: exhibit *overthinking* (Chen et al., 2024) in simple reasoning tasks and *underthinking* (Wang et al., 2025) in more difficult ones. Similarly, we observe that the same issue arises in tool usage scenarios. Current LLMs exhibit ***over-tool-reliance***, invoking tools even when tasks could be completed independently, while also exhibiting ***overconfidence*** by refusing to use tools when necessary. This inconsistency mirrors the challenges faced by O1-like models, undermining the model's tool intelligence and increasing task completion costs in real-world scenarios.

In this work, we aim to improve how LLMs decide when and how to use external tools for task completion. The main challenge is aligning the model's behavior with its knowledge boundaries, allowing it to determine when a tool is needed based on its confidence. Instead of treating the model's knowledge as simply "known" or "unknown" (Yang et al., 2023c), we propose a more nuanced approach that accounts for uncertainty. This approach recognizes an "***uncertain region***" where the model assigns probabilistic estimates to its knowledge, enabling better decision-making that balances task success and tool usage costs.

We introduce an alignment framework for efficient tool calling that combines probabilistic knowledge boundary estimation with dynamic decision-making. Our approach has two main components: 1) **Knowledge Boundary Estimation**: we propose two methods to assess the model's knowledge: consistency-based estimation based on agreement and using external ground truth to evaluate the average accuracy of multiple model samplings. 2) **Knowledge Boundary Modeling**: we construct different data to exhibit *implicit modeling*, where the model makes decisions based on predefined thresholds of knowledge certainty, and *explicit modeling*, where the model outputs both an answer and a confidence score. This framework helps the model use tools more efficiently, invoking them only when necessary, thus improving performance while reducing costs. Our approach is evaluated across multiple tool-use scenarios, demonstrating a significant reduction in unnecessary tool invocation and an improvement in overall tool efficiency. Our contributions can be summarized as follows:

- We propose a multi-objective alignment framework for efficient tool invocation, along with corresponding evaluation metrics.
- We propose the tool alignment algorithms and corresponding data generation methods.
- We conduct extensive experiments across multiple tool invocation scenarios, demonstrating the effectiveness of our approach.

2 Related Work

2.1 LLM Alignment

LLM alignment seeks to train language models to act in accordance with the user's intent, utilizing methods such as supervised fine-tuning (Wei et al., 2022; Chung et al., 2022; Zhang et al., 2023), direct preference optimization (DPO) (Rafailov et al., 2024), or reinforcement learning from human feedback (RLHF) (Stiennon et al., 2020; Ouyang et al., 2022; Glaese et al., 2022). Most works focus on enhancing the instruction-following capabilities (Sanh et al., 2021; Wei et al., 2022), helpfulness (Ding et al., 2023; Xu et al., 2023), harmlessness (Solaiman and Dennison, 2021; Bender et al., 2021), and honesty (Cui et al., 2023; Yang et al., 2023b) of LLMs. In addition, some works proposed aligning models with their knowledge boundaries (Xu et al., 2024b; Yang et al., 2023c), specifically by training LLMs to reject unknown questions. However, these approaches assume a binary view of the model's knowledge boundary—either the model knows the answer or it does not. In contrast, our work posits that knowledge boundaries are more nuanced and exist within a gray area. We propose dynamically determining the model's behavior within this ambiguous region, depending on the specific application scenario.

2.2 Tool Learning

Recent advancements in tool learning have enabled LLMs to effectively integrate external tools, enhancing real-time knowledge retrieval, multimodal functionalities, and domain-specific expertise (Yang et al., 2023a; Gupta and Kembhavi, 2023; Jin et al., 2024). Methods range from leveraging in-context learning for tool descriptions and demonstrations (Hsieh et al., 2023) to explicit training on tool-enriched datasets (Patil et al., 2023; Tang et al., 2023; Qin et al., 2023). Some works have also investigated how to accomplish tasks within a limited number of tool invocations (Zheng et al., 2024) and how to call tools more reliably (Xu et al., 2024a; Gui et al., 2024). However, previous research on tool invocation has largely overlooked the correlation between tool usage and the model's knowledge boundaries. Additionally, there has been no unified evaluation metric proposed for assessing efficient tool invocation.

3 Problem Formulation

3.1 LLM Alignment

With the rapid development of large language models (LLMs), ensuring their alignment with human instructions, preferences, and values has become a crucial research area (Wang et al., 2024). Alignment approaches are designed to optimize model responses based on predefined objectives such as helpfulness, truthfulness, and safety. Specifically, given an input prompt x_i and an alignment goal *helpfulness*, we employ the following scoring principle to represent the alignment objective:

$$s(x, y_h) > s(x, y_u), \tag{1}$$

where y_h and y_u represent a helpful response and an unhelpful response, respectively. The preference order can be determined through human annotation (Ouyang et al., 2022) or a scoring model (Gao et al., 2023a) trained with human preference data. The collected preference data can be further leveraged to train reward models or fine-tune LLM policies, thereby improving alignment with human expectations.

3.2 Multi-Objective Alignment for Efficient Tool Calling

While alignment with helpfulness is essential, efficient tool calling introduces additional alignment challenges. A well-aligned LLM should not only provide helpful responses but also minimize unnecessary tool usage, as excessive tool calls increase inference latency and computational costs. Therefore, we propose a multi-objective alignment framework that balances *helpfulness* and *tool cost*.

First, we define alignment objectives separately for helpfulness and tool cost. The helpfulness alignment objective follows:

$$s(x, y_c) > s(x, y_w), \tag{2}$$

where y_c represents a correct response, and y_w represents an incorrect response. Simultaneously, for tool cost, we define:

$$s(x, y_n) > s(x, y_t), \tag{3}$$

where y_n represents a response without tool usage, and y_t represents a response with tool usage. Combining these two objectives, our final alignment formulation becomes:

$$s(x, y_{nc}) > s(x, y_{tc}) > s(x, y_{nw}) > s(x, y_{tw}), \tag{4}$$

where y_{nc}, y_{tc}, y_{nw}, y_{tw} represent correct responses without tool usage, correct responses with tool usage, incorrect responses without tool usage, and incorrect responses with tool usage, respectively. This ordering reflects the principle that an ideal LLM should solve problems independently whenever possible, resorting to tool usage only when necessary, while also avoiding incorrect answers and unnecessary tool calls.

3.3 Evaluation Methodology

To quantify the tradeoff between helpfulness and tool cost, we define a **benefit-cost utility** function as follows:

$$u(y) = \mathbb{1}_{helpfulness}(y) - \alpha \cdot \mathbb{1}_{cost}(y), \tag{5}$$

where $\mathbb{1}_{helpfulness}(y)$, $\mathbb{1}_{tool}(y)$ equal to 1 when the response y is correct or contains tool calling, respectively. α represents the cost associated with tool usage. The overall utility of a model on a dataset with N samples is then computed as:

$$\text{Utility} = \frac{1}{N}\sum_{i=1}^{N} u(y_i) = \text{Acc} - \alpha \cdot \text{TR}, \tag{6}$$

where Acc and TR represent the overall accuracy and tool usage ratio on the dataset, respectively.

The parameter α is crucial, as it determines the relative penalty of tool usage. A larger α indicates a higher sensitivity to cost or a greater penalty for invoking tools. If α is too high, the model may completely avoid tool usage, even when necessary. Conversely, if α is too low, the model may overuse tools. Therefore, selecting a moderate α ensures a balanced tradeoff between efficiency and effectiveness. Furthermore, the cost of tool usage varies across different tasks and tools. To account for these differences, α can be set dynamically based on the specific tool being used. Empirically, in our study, we assign α values of 0.2, 0.4, and 0.6 to calculators, search engines, and external LLM reasoning, respectively. The different α values reflect the increasing computational cost and inference latency associated with these tools.

4 Methodology

4.1 Framework for Efficient Tool Learning

The key to enabling efficient tool calling lies in aligning LLMs with their own knowledge boundaries. Unlike a binary classification of knowledge into "known" and "unknown," human cognition—and by extension, LLMs—operates within a spectrum. As shown in the left part of Figure 2, there exists a large "***uncertain region***" where the model can only assign a probabilistic estimate to its knowledge. Previous works that enforce a strict

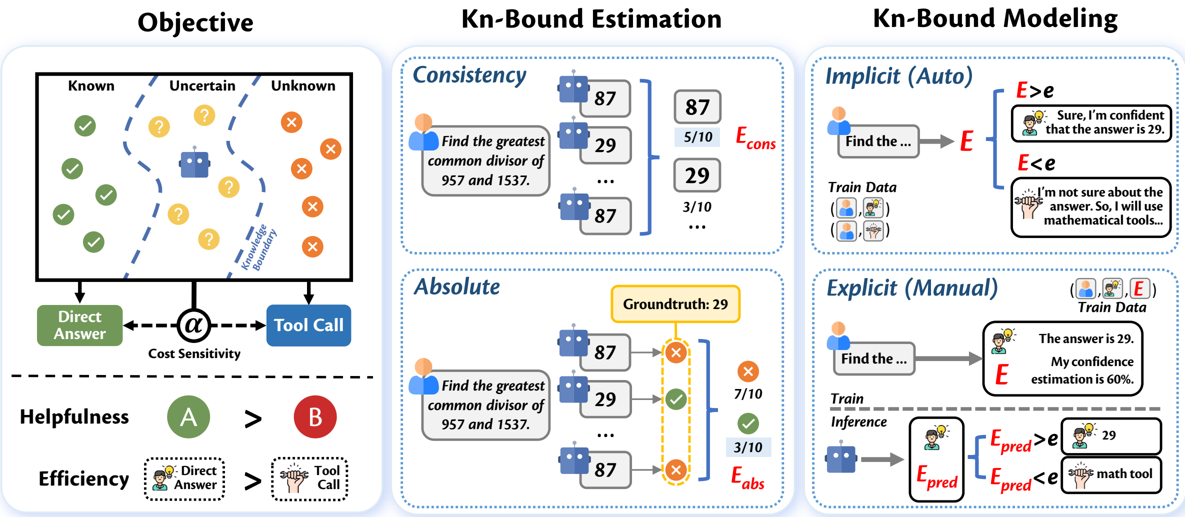

Figure 2: The overall pipeline of knowledge boundary modeling methods.

binary classification fail to capture this nuanced understanding, leading to inaccurate estimations and suboptimal tool invocation strategies.

To achieve effective tool use, the model must first develop an awareness of its knowledge boundaries and then leverage this understanding to adjust its decision-making process. This perspective aligns with the efficiency objective discussed in prior sections: a model that perceives knowledge in binary terms will struggle to adjust its behavior under varying cost considerations (represented by α). If a model simply categorizes knowledge as either "known" or "unknown," it will either always invoke a tool for uncertain cases or always answer directly, ignoring cost-sensitive optimization.

We propose a solution where the model learns to estimate its knowledge uncertainty probabilistically rather than making binary classifications. This allows for greater flexibility in tool invocation. Depending on different values of α (which represent different real-world tool costs), we can train the model to dynamically adjust its behavior. This can be implemented implicitly through controlled training data distributions or explicitly by having the model output confidence estimates that can be thresholded at inference time to determine whether a tool should be invoked.

4.2 Estimating Knowledge Boundaries

We propose two methods for knowledge boundary estimation as shown in the middle part of Figure 2:

Consistency-Based Estimation This method relies on self-consistency. We assume that if a model produces highly consistent outputs across multiple samples for a given question, it possesses a stronger grasp of the underlying knowledge. To operationalize this, we measure the variance in the model's sampled responses and use it as an indicator of knowledge certainty. Higher consistency implies greater confidence in the model's knowledge.

Absolute Estimation via Ground Truth While consistency-based estimation is useful, it does not directly leverage external validation. To address this, we introduce an absolute estimation method based on ground truth correctness. We repeatedly sample model responses for the same question and compute the average accuracy using ground truth. This provides an externally validated measure of the model's knowledge, correcting for potential biases in self-estimation.

4.3 Training Approaches

To integrate knowledge boundary estimation into the model's behavior, we employ two SFT strategies as shown in the right part of Figure 2: implicit modeling and explicit modeling.

Implicit Modeling In this approach, the model is trained to directly output actions (either answering directly or invoking a tool) based on pre-defined decision rules. Specifically, we sort all training samples based on their estimated knowledge scores and set a threshold: samples above this threshold are labeled for direct answering, while those below it are labeled for tool invocation. Since different values of α correspond to different tool usage preferences, we train separate SFT models with varying thresholds to adapt to different scenarios. This method is efficient during inference, as the model only needs to generate a single response per query. However, it requires multiple rounds of training for different values of α.

Explicit Modeling Unlike implicit modeling, explicit modeling trains the model to output both an answer and an associated knowledge confidence score. This allows dynamic adjustment of tool invocation decisions at inference time without requiring separate SFT models for different α values. During inference, we set a threshold on the confidence score: if the score is above the threshold, the model answers directly; otherwise, it invokes a tool. This approach eliminates the need for retraining but introduces additional inference latency, as each query requires both an answer and an uncertainty estimation before deciding whether to use a tool.

Each method has its advantages and drawbacks. Implicit Modeling has Faster inference (single response generation) but requires multiple training runs for different α values. Explicit Modeling is more flexible at inference time (threshold tuning without retraining) but slower due to the two-step generation process. In our experiments, we evaluate both approaches to determine the most effective strategy for efficient tool calling.

5 Experiments

5.1 Experiment Setup

5.1.1 Task Scenarios

We evaluate our approach across three scenarios, each requiring a specific external tool: symbolic computation via a calculator, factual retrieval using a retrieval-augmented generation (Gao et al., 2023b) system, and complex reasoning with a strong reasoning model. See Appendix D for more detailed experimental setup.

Arithmetic Computation (Calculator). To evaluate mathematical computation capabilities, we construct an arithmetic dataset following Liu and Low (2023). Input numbers are sampled on a logarithmic scale to ensure diverse magnitudes with minimal duplication. To enhance linguistic diversity, we use hundreds of instruction templates generated by ChatGPT. Computation is performed using a symbolic calculator as a tool, implemented via code execution for precise mathematical evaluation.

Knowledge-based QA (Retrieval-Augmented Generation). To evaluate factual knowledge retrieval, we use TriviaQA (Joshi et al., 2017), a widely used question-answering dataset. We sample 10,000 instances for training and use the 11,313 instance development set for evaluation, as the official test set ground truth is unavailable. To enhance factual accuracy, we integrate a retrieval system, leveraging Pyserini (Lin et al., 2021)—a Python toolkit designed for reproducible information retrieval with sparse and dense representations.

Complex Reasoning (Reasoning Model). To evaluate multi-step reasoning tasks, we use the MATH dataset (Hendrycks et al., 2021) with its original train-test split. Given the inherent complexity of mathematical reasoning, we employ DeepSeek-R1 (DeepSeek-AI et al., 2025) as a tool for reasoning, leveraging its strong problem-solving capabilities. However, this comes at a trade-off: higher computational cost and slower inference speed.

5.1.2 Baselines

The baseline methods are categorized into two major groups: *Prompt-based* and *Uncertainty-based*. All prompts used are listed in Appendix F.

Prompt-based Prompt-based methods govern how the model interacts with external tools and determines its tool usage behavior. The Baseline (w/o tool) approach has the model answer queries entirely on its own, relying only on internal knowledge. The Baseline (all tool) forces the model to always invoke a tool. The Auto tool method allows the model to decide when to use a tool based on its estimated confidence. ICL tool (10-shot) provides the model with 10 example interactions (5 correct, 5 incorrect) to better guide its decision on whether to answer directly or use a tool. These baselines are newly designed to reflect intuitive tool-use strategies under varying assumptions of tool accessibility and cost (see Appendix A for details).

Uncertainty-based. Uncertainty-based methods estimate the confidence of model-generated answers, which we leverage to determine the optimal utility by searching for the best confidence threshold. We explore four approaches: Raw logits (Lyu et al., 2024), P(True) (Kadavath et al., 2022), Verbalized Confidence (Tian et al., 2023), and Agreement (Self-Consistency) (Lyu et al., 2024), each providing a different way to assess model confidence (see Appendix B for details).

5.1.3 Training Details

We use two baseline models: LLAMA-3.1-8B-INSTRUCT and QWEN-2.5-7B-INSTRUCT.To align with our experimental setup, we customize the DeepSpeed-Chat (Yao et al., 2023) framework. The training process adopts a learning rate of 5×10^{-5} and a batch size of 128. All other training parameters are set to the default parameters in DeepSpeed-Chat. By default, 10,000 samples are used for Supervised Fine-Tuning. All models undergo training for 2 epochs on A800 GPUs (see Appendix C for more details).

5.2 Main Results

Table 1 compares the performance of all evaluated methods. Our approach achieves the highest *utility* scores across three scenarios, demonstrating its effectiveness in balancing task success and tool efficiency. Among our methods, Absolute-based knowledge boundary estimation outperforms Consistency-based estimation, as external supervision via ground truth labels enables more accurate boundary estimation and better tool invocation decisions. Our approach maintains accuracy comparable to the best methods while reducing tool usage by nearly 50% compared to fully automatic baselines. It also matches the Baseline (All Tools) in accuracy while significantly lowering reliance on external tools, reducing computational costs. Our training-based method further enhances efficiency compared to Auto Tool, achieving better performance while reducing tool usage. This validates the effectiveness of refining tool invocation alignment with the model's internal knowledge boundary.

5.3 Overconfidence and Over-tool-reliance

We analyze how implicit modeling shape model behavior by adjusting the SFT data ratio, which represents the proportion of training samples with tool invocation. As this ratio increases, the model's confidence estimation and reliance on external tools shift. Figure 3 illustrates how the SFT data ratio influences both overconfidence and over-tool-reliance. A higher SFT data ratio increases reliance on tools, leading to more tool invocation while reducing the model's overconfidence in its knowledge. Conversely, a lower SFT data ratio decreases tool reliance but increases overconfidence. Each dataset exhibits an optimal SFT data ratio, where this combined proportion is minimized, balancing model confidence and tool dependency. This turning point in Figure 3 serves as a guideline for optimal model selection. At this ratio, the model maintains a well-calibrated knowledge boundary while minimizing unnecessary tool usage.

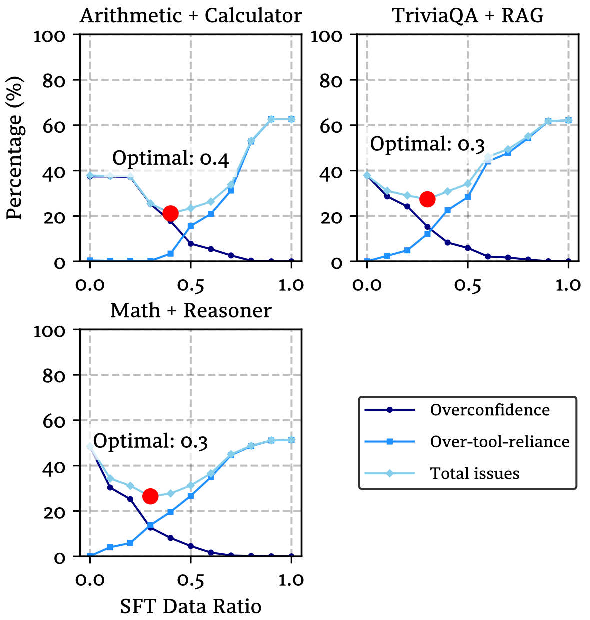

Figure 3: Trade-off between overconfidence and over-tool-reliance with different SFT data ratios.

5.4 Inference Time

Since tool invocation adds computational overhead, we assess inference cost by measuring actual execution time. Using VLLM (Kwon et al., 2023) on NVIDIA A800 GPUs (see Appendix E for detailed experimental setup), we compute per-sample inference time and aggregate the total runtime across the dataset. Figure 4 illustrates the trade-off between inference time and performance, where methods positioned towards the upper-left corner achieve a more favorable balance. Our approach consistently demonstrates superior efficiency, attaining either higher performance at the same inference time or reduce latency while maintaining accuracy. By optimizing tool usage, our method reduces computational cost while maintaining comparable performance, ensuring efficient real-world deployment and making it well-suited for practical applications.

5.5 Ablation Study

5.5.1 Implict Modeling Methods

To understand how implicit modeling affects our utility, we perform an ablation study to see how different Supervised Fine-Tuning (SFT) data ratios impact the model's behavior. The data ratio means the percentage of training examples where the model uses a tool to get the answer instead of answering on its own. We keep the total dataset size the same but change this ratio to see how it affects the model's preference for using tools or answering directly. This helps us find the best balance

Type	Method	Arithmetic + Calculator			TriviaQA + RAG			Math + Reasoner		
		Acc ↑	Tool Rate ↓	Utility(0.2) ↑	Acc ↑	Tool Rate ↓	Utility(0.4) ↑	Acc ↑	Tool Rate ↓	Utility(0.6) ↑
Llama3.1 8B										
Prompt-based	Baseline (w/o tool)	63.0	0.0	63.0	62.5	0.0	62.5	51.4	0.0	51.4
	Baseline (all tool)	99.0	100.0	79.0	95.8	100.0	55.8	96.2	100.0	36.2
	Auto tool	90.3	75.0	75.3	89.5	78.0	58.3	73.1	50.1	43.0
	ICL tool (10-shot)	91.6	62.6	79.2	85.6	69.5	57.8	53.2	4.9	50.3
Uncertainty-based	Raw logits	90.7	54.6	79.8	74.3	16.9	67.5	59.0	9.9	53.1
	P(True)	90.4	65.1	77.4	87.4	59.2	63.7	84.4	61.6	47.4
	verb. 1S top-1	65.5	7.8	63.9	77.4	32.8	64.3	64.1	16.3	54.3
	verb. 2S top-1	69.1	16.1	65.9	74.8	20.9	66.3	62.0	16.7	52.0
	agreement(consistency)	77.3	22.4	72.8	87.3	45.7	69.0	71.7	28.5	54.6
Training-based	IMPLICIT-LOGITS	80.0	33.7	73.3	74.5	24.6	64.7	85.5	56.7	51.5
	EXPLICIT-LOGITS	89.5	65.5	76.4	75.8	26.8	65.1	84.9	47.5	56.4
	IMPLICIT-CONSISTENCY	80.1	30.9	73.9	77.0	25.1	67.0	84.4	51.6	53.6
	IMPLICIT-ABSOLUTE	96.7	45.2	**87.7**	91.1	42.3	**74.2**	93.1	55.5	**59.8**
	EXPLICIT-CONSISTENCY	90.7	61.7	78.4	76.9	25.9	66.5	84.1	45.7	56.7
	EXPLICIT-ABSOLUTE	93.3	33.8	86.5	82.9	29.7	71.0	79.5	35.6	58.1
Qwen2.5 7B										
Prompt-based	Baseline (w/o tool)	67.0	0.0	67.0	51.1	0.0	51.1	74.9	0.0	74.9
	Baseline (all tool)	99.0	100.0	79.0	94.7	100.0	54.7	96.2	100.0	36.2
	Auto tool	95.7	83.4	79.0	90.4	89.6	54.6	77.1	24.5	62.4
	ICL tool (10-shot)	91.2	32.9	84.6	74.5	33.8	61.0	75.1	1.8	74.0
Uncertainty-based	Raw logits	95.1	47.8	85.5	86.6	61.7	61.9	86.9	34.1	66.4
	P(True)	94.2	63.4	81.5	79.1	53.1	57.9	86.0	30.7	67.6
	verb. 1S top-1	68.9	4.9	67.9	81.2	55.9	58.8	75.6	6.9	71.5
	verb. 2S top-1	78.9	22.4	74.4	79.5	51.5	58.9	83.9	20.2	71.8
	agreement(consistency)	91.6	22.4	87.1	86.2	47.9	67.0	97.8	38.6	74.6
Training-based	IMPLICIT-LOGITS	83.9	22.8	79.3	81.3	56.1	58.9	91.9	52.9	60.2
	EXPLICIT-LOGITS	84.2	27.2	78.8	83.2	60.1	59.2	92.9	53.9	60.6
	IMPLICIT-CONSISTENCY	82.7	17.2	79.3	84.2	58.1	61.0	96.9	54.9	64.0
	IMPLICIT-ABSOLUTE	97.6	37.9	90.1	90.7	59.1	67.1	93.9	29.0	76.5
	EXPLICIT-CONSISTENCY	90.7	61.7	78.4	72.9	23.9	63.3	89.9	22.3	76.5
	EXPLICIT-ABSOLUTE	97.3	28.8	**91.5**	80.3	30.3	**68.2**	90.1	21.2	**77.4**

Table 1: Performance comparison on three tool calling scenarios. The utility is the overall evaluation metric of accuracy and tool rate. A larger α indicates a higher cost sensitivity and a greater penalty for invoking tools.

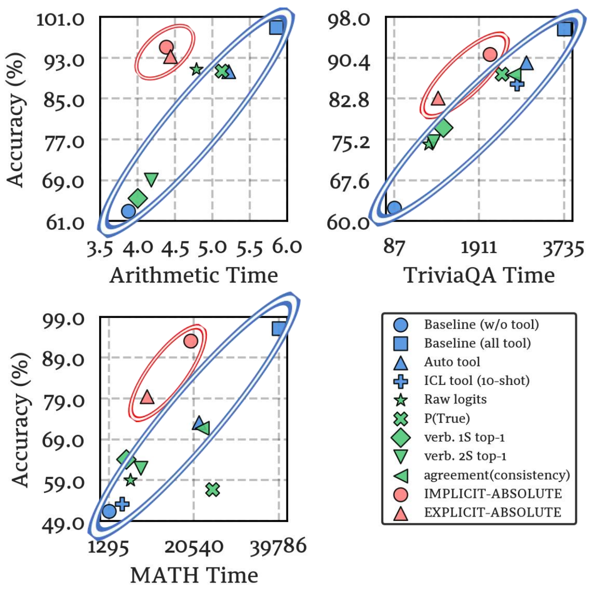

Figure 4: Performance vs. inference time (seconds).

based on cost. When using a tool is cheap, a higher ratio makes the model use tools more often, which improves accuracy by using external resources. On the other hand, if tool usage is expensive, a lower ratio makes the model answer questions independently, reducing costs. The key is to find the right balance so the model efficiently decides when to use tools based on the situation. Figure 5 shows how the data ratio affects the model's utility. At first, utility increases as the ratio goes up, reaching a peak before dropping. The best ratio is different for each dataset and depends on how much the tool costs. If tool costs are high, the optimal ratio is lower. This shows that our implicit modeling approach helps the model make smart choices based on task costs, balancing accuracy and efficiency.

5.5.2 Explicit Modeling Methods

Unlike implicit approaches, explicit modeling allows the model to directly output confidence scores alongside its predictions, enabling threshold-based decision-making for tool invocation. To further evaluate its effectiveness, we compare explicit modeling with uncertainty-based baselines, as both methods fundamentally rely on confidence estimation to determine knowledge boundaries. To ensure a fair comparison, we adjust the confidence threshold to control the tool invocation ratio, systematically varying the threshold to assess model performance at different levels of tool usage. As shown in Figure 6 illustrates the relationship between tool invocation rate and model performance across various confidence thresholds. Explicit modeling consistently outperforms uncertainty-based baselines at all invocation ratios, demonstrating its ability to provide a more reliable estimation of knowledge boundaries. The performance gap remains stable, highlighting the robustness of explicit confidence modeling. By leveraging these confi-

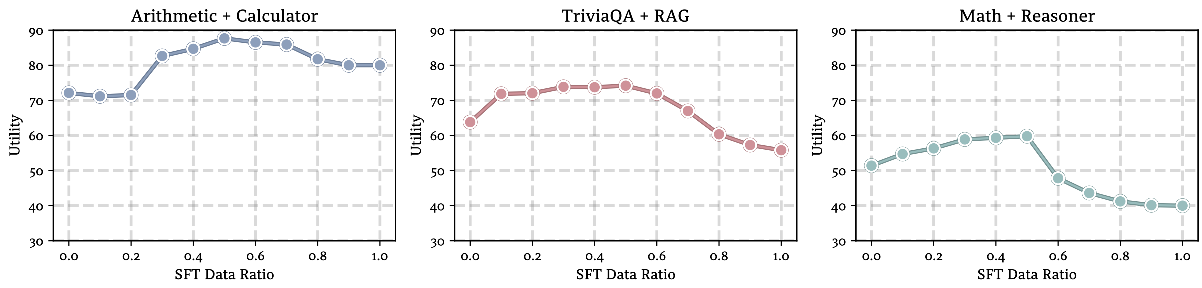

Figure 5: **Effect of SFT Data Ratio on Utility.** The ratio represents the proportion of training samples in which the model invokes a tool rather than answering directly.

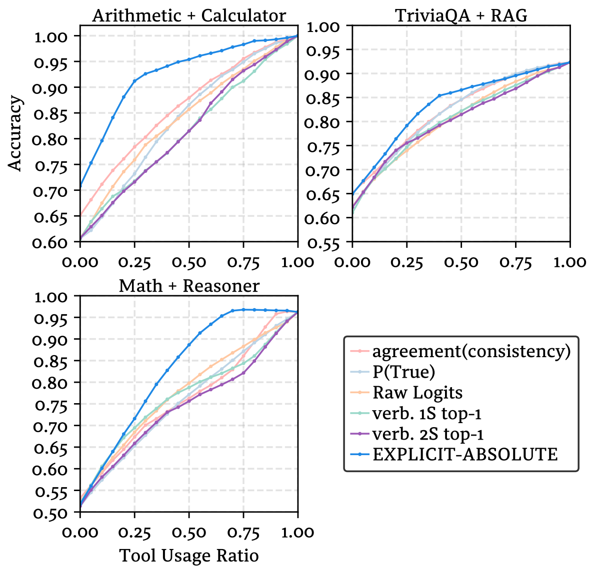

Figure 6: Comparison of tool invocation strategies: explicit modeling vs. uncertainty-based baselines.

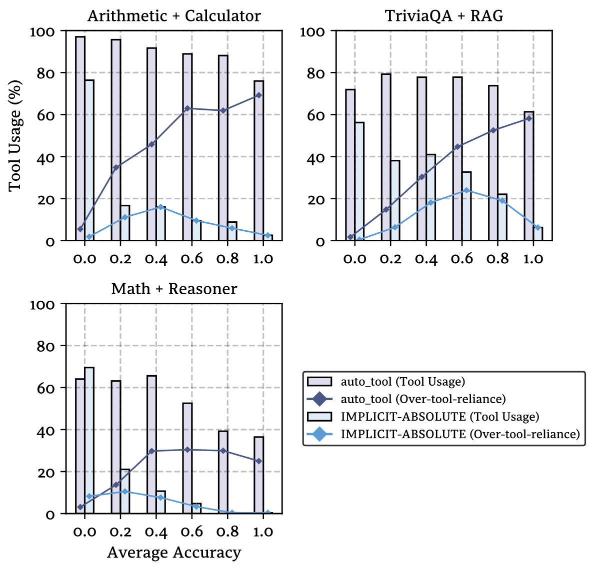

Figure 7: Comparison of tool usage and over-tool-reliance across different accuracy levels.

dence scores, our approach enables finer control over tool invocation, optimizing task success while reducing unnecessary computational overhead.

5.6 Knowledge Boundary Alignment

To examine whether the model learns about knowledge boundary, we compare our method with `auto_tool` in terms of tool invocation distribution. Figure 7 presents tool usage across different accuracy levels. Higher accuracy reflects a better understanding of the problem. An ideal model should rely on tools for challenging cases while minimizing tool use for confidently answered questions. However, `auto_tool` exhibits a nearly uniform tool invocation pattern, suggesting it lacks awareness of its knowledge boundaries. In contrast, our method shows a gradual decline in tool usage as accuracy increases, indicating adaptive tool invocation based on knowledge confidence. We also analyze over-tool-reliance, where the model uses tools unnecessarily despite being capable of answering correctly. Figure 7 shows that the baseline exhibits increasing over-tool-reliance with accuracy, leading to unnecessary computational overhead. Conversely, our method reduces over-tool-reliance, enabling more intelligent invocations.

6 Conclusion

In this work, we introduced a novel approach to improve LLMs' decision-making regarding when and how to use external tools. By incorporating the concept of an "uncertain region" and probabilistic knowledge boundary estimation, our framework enables more informed and efficient tool usage. Through extensive experiments, we demonstrated that our approach reduces unnecessary tool calls, improving performance and cost-effectiveness. By combining implicit and explicit modeling techniques, we provide the model with greater flexibility in real-time decisions. Our work advances LLMs' tool intelligence, ensuring more judicious and efficient tool invocation. Future work can explore further refinements and broader applications.

Limitations

This work primarily proposes an alignment framework for efficient tool invocation, evaluated through experiments on three datasets. On the one hand, the number of tools used in these experiments is limited, with a selection of three representative tools: a mathematical calculator, a search engine, and an external large model. This choice is motivated by the fact that most tools possess highly specific knowledge. For example, tools that retrieve weather information for a particular day contain knowledge that does not overlap with that of the model, requiring the model to invoke the tool to complete the task. On the other hand, different models and knowledge sources can also be framed as tools, meaning that the discussion in this work on modeling knowledge boundaries remains highly valuable. In addition, the experiments in this work were conducted on only two open-source models, as obtaining baseline data for closed-source models presents significant challenges. For instance, methods such as uncertainty estimation often require access to specific token logits, which are difficult to obtain for proprietary models. This limitation affects the generalizability of the experimental results, as the performance of closed-source models may differ in ways that cannot be captured without direct access to their internals.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*.

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, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing Chat Language

Models by Scaling High-quality Instructional Conversations. *ArXiv preprint*, abs/2305.14233.

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*.

Leo Gao, John Schulman, and Jacob Hilton. 2023a. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Gemini Team. 2023. Gemini: A Family of Highly Capable Multimodal Models. *Preprint*, arXiv:2312.11805.

Amelia Glaese, Nat McAleese, Maja Trebacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. 2022. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*.

Anchun Gui, Jian Li, Yong Dai, Nan Du, and Han Xiao. 2024. Look before you leap: Towards decision-aware and generalizable tool-usage for large language models. *arXiv preprint arXiv:2402.16696*.

Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual programming: Compositional visual reasoning without training. In *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2023)*, pages 14953–14962.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings. *ArXiv preprint*, abs/2305.11554.

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. *NeurIPS*.

Cheng-Yu Hsieh, Si-An Chen, Chun-Liang Li, Yasuhisa Fujii, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2023. Tool documentation enables zero-shot tool-usage with large language models. *ArXiv preprint*, abs/2308.00675.

Qiao Jin, Yifan Yang, Qingyu Chen, and Zhiyong Lu. 2024. GeneGPT: augmenting large language models with domain tools for improved access to biomedical information. *Bioinformatics*, 40(2):btae075.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2356–2362.

Tiedong Liu and Bryan Kian Hsiang Low. 2023. Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks. *arXiv preprint arXiv:2305.14201*.

Qing Lyu, Kumar Shridhar, Chaitanya Malaviya, Li Zhang, Yanai Elazar, Niket Tandon, Marianna Apidianaki, Mrinmaya Sachan, and Chris Callison-Burch. 2024. Calibrating large language models with sample consistency. *arXiv preprint arXiv:2402.13904*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large Language Model Connected with Massive APIs. *ArXiv preprint*, abs/2305.15334.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. *Preprint*, arXiv:2307.16789.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. *ArXiv preprint*, abs/2302.04761.

Irene Solaiman and Christy Dennison. 2021. Process for adapting language models to society (palms) with values-targeted datasets. *Advances in Neural Information Processing Systems*, 34:5861–5873.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023. ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases. *Preprint*, arXiv:2306.05301.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. 2023. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *arXiv preprint arXiv:2305.14975*.

Xinpeng Wang, Shitong Duan, Xiaoyuan Yi, Jing Yao, Shanlin Zhou, Zhihua Wei, Peng Zhang, Dongkuan Xu, Maosong Sun, and Xing Xie. 2024. On the essence and prospect: An investigation of alignment approaches for big models. *arXiv preprint arXiv:2403.04204*.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models are Zero-Shot Learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. WizardLM: Empowering Large Language Models to Follow Complex Instructions. *Preprint*, arXiv:2304.12244.

Hongshen Xu, Su Zhu, Zihan Wang, Hang Zheng, Da Ma, Ruisheng Cao, Shuai Fan, Lu Chen, and Kai Yu. 2024a. Reducing tool hallucination via reliability alignment. *arXiv preprint arXiv:2412.04141*.

Hongshen Xu, Zichen Zhu, Da Ma, Situo Zhang, Shuai Fan, Lu Chen, and Kai Yu. 2024b. Rejection improves reliability: Training llms to refuse unknown questions using rl from knowledge feedback. *arXiv preprint arXiv:2403.18349*.

Linyao Yang, Hongyang Chen, Zhao Li, Xiao Ding, and Xindong Wu. 2023a. ChatGPT is not Enough: Enhancing Large Language Models with Knowledge Graphs for Fact-aware Language Modeling. *ArXiv preprint*, abs/2306.11489.

Yuchen Yang, Houqiang Li, Yanfeng Wang, and Yu Wang. 2023b. Improving the reliability of large language models by leveraging uncertainty-aware in-context learning. *arXiv preprint arXiv:2310.04782*.

Yuqing Yang, Ethan Chern, Xipeng Qiu, Graham Neubig, and Pengfei Liu. 2023c. Alignment for honesty. *arXiv preprint arXiv:2312.07000*.

Zhewei Yao, Reza Yazdani Aminabadi, Olatunji Ruwase, Samyam Rajbhandari, Xiaoxia Wu, Ammar Ahmad Awan, Jeff Rasley, Minjia Zhang, Conglong Li, Connor Holmes, et al. 2023. Deepspeed-chat: Easy, fast and affordable rlhf training of chatgpt-like models at all scales. *arXiv preprint arXiv:2308.01320*.

Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, et al. 2022. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Yuanhang Zheng, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. 2024. Budget-constrained tool learning with planning. *arXiv preprint arXiv:2402.15960*.

A Prompt-based Methods

We implement four prompt-based baseline methods to facilitate a fair and interpretable comparison across varying tool-use strategies. These baselines are designed to represent intuitive and commonly used decision patterns along the spectrum of tool accessibility and reliance. Below, we provide detailed descriptions and the design rationale for each baseline.

Baseline (w/o tool) In this setting, the model is instructed to answer each question using only its internal parametric knowledge, without access to any external tool. This baseline serves to evaluate the model's raw performance without any form of external assistance. It provides a lower bound for performance, isolating the contribution of the model's internal memory. Moreover, it allows us to quantify the incremental benefits gained from tool usage and to identify scenarios where tools are essential for accurate responses.

Baseline (all tool) The model is required to always utilize external tool outputs—such as retrieved documents or calculator results—when generating an answer, irrespective of its confidence level. This baseline simulates an over-reliance on tools, representing a naïve strategy where the model defaults to tool usage regardless of necessity. It approximates an upper bound on task performance under the assumption that tool outputs are generally helpful. At the same time, it highlights the trade-off between performance and tool usage cost, particularly in settings sensitive to latency or resource constraints.

Auto Tool (Zero-Shot) The model is prompted to make a binary decision on whether to invoke a tool, based solely on its internal confidence, without any fine-tuning or in-context examples. This baseline evaluates the model's zero-shot uncertainty estimation capability and its ability to make tool-use decisions guided purely by prompt instructions. It offers a natural and competitive baseline for comparison with approaches that incorporate explicit confidence training or reinforcement.

ICL Tool (10-Shot In-Context Learning) This method extends the Auto Tool baseline by prepending 10 in-context examples (5 correct answers without tools, and 5 correct answers with tools) that demonstrate when to use or avoid tool invocation. The goal of this baseline is to assess whether the model can learn a tool-use decision policy implicitly from in-context demonstrations. By providing examples of both high-confidence (tool-free) and low-confidence (tool-required) responses, the model is expected to generalize and apply similar decision criteria to new inputs.

B Uncertainty Estimation Methods

This section provides a comprehensive overview of the uncertainty estimation techniques employed in our study. These methods aim to quantify model confidence in its predictions, helping regulate tool invocation and decision-making.

Raw Logits. This approach estimates confidence using the model's logit values, specifically by computing the exponential of the average log probability of the generated tokens. This metric is mathematically equivalent to the reciprocal of perplexity, where lower perplexity indicates higher confidence, effectively capturing how certain the model is in its prediction.

Agreement (Consistency-based). In this method, confidence is determined by measuring the proportion of generated responses that align with the most frequently predicted answer. A higher agreement percentage suggests greater internal consistency in the model's responses, thereby indicating a stronger level of confidence in its generated output.

P(True). This method involves prompting the model to explicitly assess the correctness of its own response. The confidence score is derived from the normalized probability assigned to the 'True' token, reflecting the model's self-evaluated likelihood that its answer is correct.

Verbalized Confidence: 1-Stage Top-k (Verb. 1S Top-k). In this one-stage approach, the model generates the top k candidate answers along with their respective probabilities in a single pass. The highest-ranked answer and its assigned probability serve as an indicator of confidence, offering a direct estimation of the model's certainty in its response.

Verbalized Confidence: 2-Stage Top-k (Verb. 2S Top-k). Unlike the single-stage method, this two-stage approach first prompts the model to generate multiple candidate answers and then separately assigns probabilities to each of them in a second inference step. The final confidence score is computed based on these probabilities, allowing for a refined estimation that accounts for potential self-correction.

These uncertainty estimation techniques play a crucial role in calibrating tool invocation decisions, ensuring that external tools are utilized effectively based on the model's confidence in its own predictions. To optimize utility, we sort all confidence scores across responses and use each unique score as a potential threshold, systematically evaluating its impact on tool invocation.

C Training Details

We use two baseline models: LLAMA-3.1-8B-INSTRUCT and QWEN-2.5-7B-INSTRUCT.To align with our experimental setup, we customize the DeepSpeed-Chat (Yao et al., 2023) framework. The training process adopts a learning rate of

5×10^{-5} and a batch size of 128. All other training parameters are set to the default parameters in DeepSpeed-Chat. By default, 10,000 samples are used for Supervised Fine-Tuning. All models undergo training for 2 epochs on A800 GPUs.

We train our models by leveraging the confidence scores estimated from the aforementioned methods. Specifically, the model is trained using these different confidence estimation strategies—LOGITS, CONSISTENCY, and ABSOLUTE—as supervisory signals to guide and calibrate its learning process.

LOGITS This approach estimates confidence using the model's logit values, specifically by computing the exponential of the average log probability of the generated tokens. This metric is mathematically equivalent to the reciprocal of perplexity, where lower perplexity indicates higher confidence, effectively capturing how certain the model is in its prediction.

CONSISTENCY In this method, confidence is determined by measuring the proportion of generated responses that align with the most frequently predicted answer. A higher agreement percentage suggests greater internal consistency in the model's responses, thereby indicating a stronger level of confidence in its generated output.

ABSOLUTE This method estimates the model's confidence by measuring the proportion of generated responses that align with external supervision (i.e., the ground-truth labels). It uses external signals to calibrate the model's confidence.

D Experimental Setup

Arithmetic Computation. For arithmetic tasks, we use a dataset consisting of 10,000 training samples and 1,000 test samples. To ensure the quality of generated arithmetic expressions, we filter out any syntactically incorrect or malformed expressions that do not conform to standard arithmetic formats. Symbolic computation is performed using the SymPy library, which provides a robust framework for symbolic mathematics and equation evaluation.

Knowledge-based QA (TriviaQA). For knowledge-based question answering, we randomly select 10,000 training instances from the full TriviaQA training set. The retrieval system is employed only during inference and does not participate in training. During training, the model is only exposed to the tool invocation format, but actual retrieval is not performed. We follow the Pyserini setup for TriviaQA and utilize a sparse retriever to retrieve the top 100 highest-scoring passages. To improve retrieval accuracy, we further filter passages that contain the correct answer and refine the selection using ChatGPT, eliminating irrelevant noisy passages. This ensures that the retrieved information is reliable, preventing erroneous tool invocation from negatively impacting final performance.

Complex Reasoning (MATH). For mathematical problem-solving, we process the MATH dataset following its original settings. We utilize a total of 7500 training samples and 5000 test samples, adhering strictly to the dataset's official evaluation protocol to ensure consistency and comparability with prior work. We employ DeepSeek-R1 (671B) as the external reasoning model, deploying it locally using VLLM on a cluster of 32 NVIDIA A800 GPU. The model operates in a zero-shot setting. To mitigate excessive inference latency, we instruct the model to generate concise responses while maintaining reasoning completeness. Despite this constraint, DeepSeek-R1 still significantly surpasses our primary models in response time.

E Inference Time Experimental Setup

For inference time evaluation, we employ the VLLM framework and conduct experiments on two NVIDIA A800 GPUs. To obtain a precise measurement of raw inference latency, we process input samples sequentially, without applying any parallelization techniques such as batching. We measure only the pure inference time, excluding any overhead from data loading. All other parameters remain at their default settings, and the model is loaded in bfloat16 format to optimize memory usage while preserving numerical precision.

F Prompts Used in Experiments

F.1 Prompts Used in Different Prompt-based Methods

The prompts used for different datasets are presented in the following sections. Table 2 shows the prompts for the **MATH dataset**, Table 3 contains the prompts for the **Arithmetic dataset**, and Table 4 presents the prompts for the **TriviaQA dataset**.

F.1.1 Prompts for MATH Dataset

Table 2 lists the prompts used for different methods when evaluating the MATH dataset.

F.1.2 Prompts for Arithmetic Dataset

Table 3 lists the prompts used for different methods when evaluating the Arithmetic dataset.

F.1.3 Prompts for TriviaQA Dataset

Table 4 lists the prompts used for different methods when evaluating the TriviaQA dataset.

F.2 Prompts Used in Different Uncertainty-based Methods

The prompts are shown in Table 5.

F.3 Question Templates

The examples of arithmetic question templates are shown in 6.

Baseline (w/o tool) - MATH

Given the following problem, break it down into steps and reason through each part before arriving at a final conclusion. Your final answer MUST be enclosed in \boxed{}.
Problem: {question}

Baseline (all tool) - MATH

Given the following problem, break it down into steps and reason through each part before arriving at a final conclusion. Your final answer MUST be enclosed in \boxed{}.
Problem: {question}

Auto Tool - MATH

Given the following problem. If you can solve it directly with confidence, your final answer must be in \boxed{} format. If you cannot solve it directly, call the tool immediately without reasoning, using this format:
{{
  "tool_name": "math_solver"
}}
Problem: {question}

ICL Tool (10-shot) - MATH

Given the following problem. If you can solve it directly with confidence, your final answer must be in \boxed{} format. If you cannot solve it directly, call the tool immediately without reasoning, using this format:
{{
  "tool_name": "math_solver"
}}
Examples: {example}
Problem: {question}

Table 2: Prompts Used in Different Methods for MATH Dataset.

Baseline (w/o tool) - Arithmetic

Given the following problem, provide the final answer directly.
Problem: {question}
Your response should only be "The final answer is [answer]" where [answer] is the response to the problem.

Baseline (all tool) - Arithmetic

Use a calculator to solve the question. Format your output as a JSON object in the following structure:
{{
  "calculator": "<expression>"
}}
Problem: {question}

Auto Tool - Arithmetic

If you are confident in your answer, output the final answer directly. If unsure, use the calculator tool and respond with a JSON object formatted as:

{{
  "tool_name": "calculator"
}}
Problem: {question}

ICL Tool (10-shot) - Arithmetic

If you are confident in your answer, output the final answer directly. If unsure, use the calculator tool and respond with a JSON object formatted as:

{{
  "tool_name": "calculator"
}}

Examples: {example}
Problem: {question}

Table 3: Prompts Used in Different Methods for Arithmetic Dataset.

Baseline (w/o tool) - TriviaQA
Answer the following question. Your response should only be "The final answer is [answer]" where [answer] is the response to the problem. Problem: {question}
Baseline (all tool) - TriviaQA
{documents} Based on the information in this document, answer the following question accurately. Problem: {question}
Auto Tool - TriviaQA
Answer the following question directly if you are confident in your knowledge. If you are uncertain or need to retrieve information, respond with a JSON object in the following format: {{ "tool_name": "search_info" }} Problem: {question}
ICL Tool (10-shot) - TriviaQA
Answer the following question directly if you are confident in your knowledge. If you are uncertain or need to retrieve information, respond with a JSON object in the following format: {{ "tool_name": "search_info" }} Examples: {example} Problem: {question}

Table 4: Prompts Used in Different Methods for TriviaQA Dataset.

Logits-based Prompt
You are a helpful assistant. Answer the following question as accurately as possible. Question: {question}
P(true) Prompt
You are a helpful assistant. You should judge whether the answer to the given question is True or False. Please only reply with a simple word "True" or "False". Answer the following questions as accurately as possible. Question: {question} Answer: {answer} Is the above answer correct? (True / False)
Consistency Prompt
You are a helpful assistant. Answer the following question as accurately as possible. Provide ONLY the direct answer without any explanation. Question: {question}
Verb. 1S top1 Prompt
You are a helpful assistant, and you are always completely honest and DIRECT in your responses. Provide a brief, concise answer along with an explicit confidence percentage (0-100%) about the correctness of your response. Question: {question}
Verb. 2S top1 Prompt
You are a helpful assistant, always completely honest and direct in your responses. You are also transparent about your confidence level and can honestly share how certain you are about the answer. Question: {question} Answer: {previous_answer} How confident are you in the above answer (0-100%)?

Table 5: Prompts Used in Uncertainy-Based Estimation Methods.

Arithmetic Question Templates
● Compute the result of {input}.
○ Answer the following question: {input}
● Determine {input}
○ Can you solve for {input}?
● Calculate {input}.
○ Help me determine the value of {input}.
● Please calculate {input}
○ Can you solve and provide the value of {input}?
● What does {input} yield?
○ Assist me in calculating {input}.
● Evaluate {input} and let me know the computed value.
○ Can you compute the value of {input}?
● Compute this: {input}.
○ Determine the numeric value resulting from {input}.
● Can you provide a stepwise solution for evaluating {input}?
○ Solve this math problem: {input}
● Compute the mathematical expression {input} and yield the result.
○ Solve this problem: {input}
● What is the value of {input}?
○ Can you tell me the result of {input}?
⋮

Table 6: Examples of arithmetic question templates generated by ChatGPT, where {input} is substituted with arithmetic questions using two randomly selected integers.