Uncertainty Calibration for Tool-Using Language Agents

Hao Liu[1] **Zi-Yi Dou**[2] **Yixin Wang**[3] **Nanyun Peng**[2] **Yisong Yue**[1]
[1]Caltech [2]University of California, Los Angeles [3]University of Michigan
{hliu3,yyue}@caltech.edu {zdou,violetpeng}@cs.ucla.edu yixinw@umich.edu

Abstract

There is increasing interest in equipping language models with the ability to leverage external tools for complex, goal-oriented tasks. However, interacting with external tools introduces inherent uncertainties due to imperfections and misalignments between the tools' outputs and the agents' internal models, often leading to suboptimal outcomes. We thus study the problem of tool-use calibration in language agents, and identify prompt design and execution trace selection as two primary areas that suffer from miscalibration. We then propose PROBECAL, which recalibrates the internal probabilities of tool-using language agents to better reflect the actual effectiveness of tool interactions, and enables a more appropriate selection of prompts and execution paths. We empirically show that PROBECAL can significantly and consistently improve off-the-shelf language models in tool-using applications.

1 Introduction

Language agents are increasingly used to perform tasks and interact with a variety of external tools to achieve specific, goal-oriented objectives (Schick et al., 2024; Hao et al., 2024). Recent advancements have focused on designing agents to more effectively utilize a diverse set of tools for different tasks (Surís et al., 2023; Lu et al., 2024; Wang et al., 2024b). Tool-using agents have also been applied to many domains, such as employing retrieval tools for multi-hop reasoning (Yang et al., 2018), leveraging visual perception models for visual question answering (Hudson and Manning, 2019), and utilizing calculators for mathematical reasoning (Hendrycks et al., 2021).

Despite these advances, tool use poses inherent uncertainties for language agents. Tools may be imperfect and produce erroneous outputs, leading to suboptimal interactions and outcomes. For instance, in tasks requiring knowledge retrieval, the quality of the retrieval tool can significantly affect the agent's performance. If the retrieval tool fails to fetch accurate or relevant information, the agent's ability to reason and generate correct responses is compromised. In visual question answering tasks, Surís et al. (2023) demonstrated that even with the correct algorithms, the performance bottleneck often lies in imperfect visual perception.

The above challenges highlight the need for systematically modeling and managing the uncertainties associated with tool-use in language agents. Existing approaches generally rely on the agent's internal probabilities to choose which tools to use and how to use them (e.g., Lu et al. (2024); Wang et al. (2024b)). However, because these tools are typically not involved during the agent's training, their operation and outputs can be misaligned with how the agent reasons using the tools' outputs.

In this paper, we study this problem of tool-use calibration, where the goal is to calibrate a language agent's internal probability estimates to more accurately reflect the actual success rates of tool-using decisions. As illustrated in Figure 1, we identify two primary scenarios where such misalignment can be significantly mitigated through uncertainty calibration: *language agent prompts* and *tool execution traces*. Our key technical contribution is a set of calibration methods called Probing Calibration (PROBCAL), which draws inspiration from probing literature (Elazar et al., 2021; Belinkov, 2022), involves training a classifier using large language model embeddings to predict the expected reward for a given prompt or execution trace. We also explore other design choices such as reweighting during training and temperature scaling (Guo et al., 2017) during inference to further refine the calibration of language agents, and provide a systematic analysis of different techniques.

We conduct experiments using various types of tool-using language agents, including both *static* and *dynamic* agents, depending on whether a fixed

Findings of the Association for Computational Linguistics: EACL 2024, pages 16781–16805

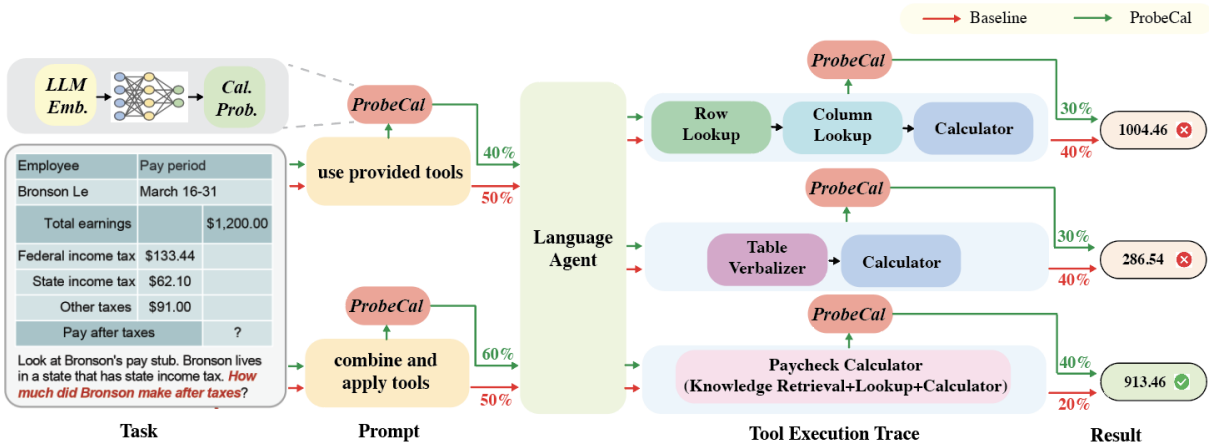

Figure 1: Overview of our PROBECAL framework for calibrating tool-using language agents. The process begins with a given task, exemplified by calculating pay after taxes from a pay stub. The agent is provided with a user prompt to utilize its tool-using capabilities, generating multiple execution traces. These traces involve different combinations of tool applications, such as row lookup, column lookup, and calculations. However, integrating user prompts and external tools introduces uncertainties, often resulting in misalignment between the agent's internal confidence levels and actual success rates. The framework addresses these issues by calibrating the agent's confidence levels with actual performance, based on which we select the most suitable prompt and execution trace. The calibration model is implemented as an MLP with LLM embeddings as inputs and calibrated probabilities as outputs, and is trained with the execution result supervision. The figure is adapted from Lu et al. (2024).

or dynamic set of tools is available. Our results on the MATH dataset (Hendrycks et al., 2021) and the TabMWP dataset (Lu et al., 2022) demonstrate that off-the-shelf language models are not well-calibrated for tool-using applications. Our PROBECAL approach significantly improves both the calibration and performance of these models consistently across different tasks. For example, applying PROBECAL results in over 7% improvement on TabMWP in dynamic tool-using settings compared with strong baselines. Furthermore, our analyses reveal that our approach can generally perform well without the need for the LLM logit or posthoc calibration, such as temperature scaling, which can translate into simpler overall frameworks for tool-using language agents.

2 Tool-Use Calibration for Language Agents

Tool-Using Language Agents. We define a *tool-using language model* as an agent that can interact with external environment such as program compilers via a defined set of function interfaces $\mathcal{F} = \{f_1, \ldots, f_k\}$. Each function f_i in $\mathcal{F}$ represents a callable program that the LM can utilize to perform specific tasks or computations external to the LM itself. Given a textual query q and the toolbox $\mathcal{F}$, the language model synthesizes a program $z = \text{LM}(\pi(q, \mathcal{F}))$, where $\pi(q, \mathcal{F})$ is the prompt generated from both q and $\mathcal{F}$, and fed to the language agent; LM is the language agent being able to interpret natural language queries and generate corresponding executable code. The generated program z is then executed in the given environment e by an execution engine ϕ, such that the result of the execution is given by $r = \phi(e, z)$.

Following previous work (Surís et al., 2023; Wang et al., 2024b), the programs $z \in \mathcal{Z}$ are directly represented as Python code. The execution engine ϕ is capable of handling various types of inputs x (e.g., images, videos) and producing outputs r of different types (e.g., text, image crops). In practice, multiple plausible candidate programs $\{z_1, \ldots, z_N\}$ are generated from the language agent, with each program z_i being a tool execution trace, and decoding strategies such as beam search and self-consistency (Wang et al., 2022; Chen et al., 2023) are used to select the best program from this candidate set.

2.1 Misalignment Scenarios in Tool-Using Language Agents

The central challenge that we study is when language agents do not have a calibrated or aligned internal model of the effectiveness of tools, thereby leading to suboptimal tool-use.

Because the model is typically not trained with tools, their estimation of whether a tool-using trajectory is correct is often mis-calibrated. Consider the following question from Math Algebra

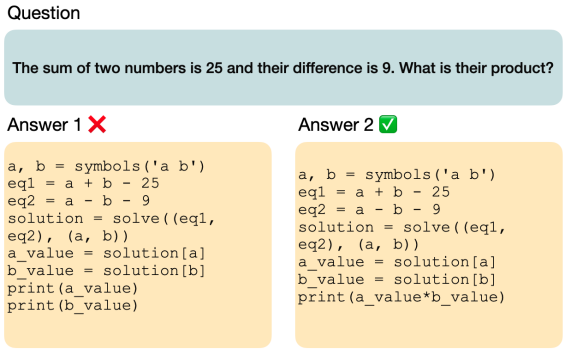

Figure 2: The LLM generates two different code sequences to answer the question, with the first one being incorrect and second one being correct. However, their textual formats are similar, resulting in similar uncertainty estimations with the uncalibrated LMs.

(Hendrycks et al., 2021): "The sum of two numbers is 25 and their difference is 9. What is their product?", the LLM (`CodeLlama-7B-Instruct-hf`) generates two different code sequences to answer this question, with the first one being incorrect and second one being correct, as shown in Figure 2. However, their textual formats are rather similar, resulting in similar uncertainty estimations with the uncalibrated LMs. We identify two primary scenarios where uncertainty misalignment can be alleviated through calibration: *language agent prompts* and *tool execution traces*. The calibration in these scenarios can significantly impact the performance and reliability of the language agents.

2.1.1 Language Agent Prompts

Language models are often equipped with tool-using capabilities through the provision of carefully designed prompts. These prompts, denoted as π, include information about the tools, such as task instructions, example pairs of queries and their corresponding solutions, and comprehensive documentation detailing the tools' functionalities. The variability in the design and content of these prompts introduces uncertainty in the performance of the language models. Formally, let $\mathcal{P}$ represent the space of possible prompts. The prompt $\pi \in \mathcal{P}$ for a given query q can vary based on factors such as the specificity of the examples, the clarity of the instructions, and the comprehensiveness of the documentation. This variability can be an area of miscalibration, as different prompts may lead to different levels of tool utilization proficiency.

Prompt Design. These prompts can include information such as task instructions, example pairs of queries and solutions involving the tools, and documentation detailing the tools' functionality. In addition, the prompts can specify if the agents (1) only use *primitive functions*, which are basic, low-level operations within the task environment, such as retrieving data from a database; (2) can induce and use composite functions, such as data aggregation or filtering, integrate multiple primitive functions into a single, cohesive operation. By defining whether the use of composite functions is allowed, prompts can guide the agents to either rely on simpler, more granular steps or leverage more advanced, integrated actions that align better with the overall task objectives.

Variability in prompt design can significantly affect the agent's performance, as different prompts may result in varying levels of understanding and application of the tools. Effective prompt design should balance between providing sufficient detail to guide the agent while avoiding overwhelming it with unnecessary complexity. Clear and concise prompts help minimize misunderstandings and reduce the likelihood of errors in task execution. Therefore, prompt design is a crucial factor in managing uncertainty and enhancing the reliability of tool-using language agents.

2.1.2 Tool Execution Traces

Another significant area of miscalibration arises from variations in tool sequencing and the availability of multiple tools that can perform the same task. For any given task, there can be numerous plausible execution traces, each represented as a sequence of tool usages. Let $\mathcal{T}$ denote the set of all possible execution traces, where each trace $z \in \mathcal{T}$ is a sequence of function calls from the toolbox $\mathcal{F}$. The challenge lies in selecting the optimal execution trace that leads to the correct solution.

Given a set of candidate execution traces $\{z_1, \ldots, z_N\}$ generated by the language model, decoding strategies such as beam search and self-consistency are employed to select the best trace (Wang et al., 2024b). However, the presence of multiple valid but suboptimal traces introduces uncertainty in the model's performance. The correct trace z^* that yields the desired output r is not always apparent, making it difficult for the language agent to consistently produce the correct solution.

Trace Selection. The process of selecting the optimal execution trace involves evaluating the plausibility and effectiveness of different sequences of tool usages. Each candidate trace must be assessed

for its potential to achieve the desired result. This evaluation can be challenging due to the inherent complexity and variability of the tasks. The language agent must not only generate multiple potential traces but also rank and choose the most promising one. This adds a layer of complexity and potential for error, as even minor deviations in the trace can lead to incorrect outcomes.

3 Probing Calibration (PROBECAL)

As discussed, the performance of tool-using language agents is influenced by the variability in tool information prompts and the complexity of determining the correct tool execution traces. In this part, we introduce our proposed methods to calibrate these uncertainties and use the calibrated scores to enhance the performance of language agents. Inspired by the literature on probing, where a separate diagnostic model is trained on top of a pretrained language model's internal representations to study linguistically-informed properties (Belinkov, 2022; Lasri et al., 2022), we employ the training of a classification model that takes inputs as the embeddings from a large language model and to predict the expected reward given a prompt or execution trace.

Reward Estimation. Given a dataset containing questions and their corresponding answers, we feed models with different candidate prompts or sample multiple execution traces to collect the corresponding embeddings from a language agent and the final reward r for each prompt or trace. For example, in question-answering settings, the reward r equals 1 if the final result matches the ground truth answer and 0 otherwise. Formally, given the embedding ϕ from the language model and the corresponding reward r, reward calibration involves training a reward classification calibration model to learn the mapping $P(r|\phi)$ from ϕ to r. In our setting, we use a three-layer MLP as the model.

We propose to use the embedding-based reward estimation to calibrate the language model, which we call Probing Calibration (PROBCAL). We apply the embedding-based reward estimation to the prompt design setting, (Prompt Calibration (PROBECAL-PROMPT)), the trace selection setting (Trace Calibration (PROBECAL-TRACE)), and both of the settings (Prompt and Trace Calibration (PROBECAL-PROMPT&TRACE)) as follows.

Prompt Selection Calibration. Under our framework, we consider the problem of calibrating the probability of selecting which prompt to choose for different questions. Formally, for each question q, we want to select each prompt $\pi \in \mathcal{P}$ proportional to the estimated reward $P(\hat{r}|\phi)$ obtained from a calibration method, which we will illustrate in the next section. This is related to a classic method for decision-making under uncertainty called Boltzmann exploration, where each action is selected proportional to the exponential of the expected reward divided by a temperature parameter (Cesa-Bianchi et al., 2017).

Execution Trace Selection Calibration. Many existing strategies rely on language agents to estimate the success probability, which can be unreliable since these tools are external to the agents. We explicitly calibrate the success probability and compute the answer by estimating the expected reward $P(r|a, z_i, q)$ using a calibration model: $\arg\max_a \mathcal{P}(a|q) = \arg\max_a \sum_{z_i} \mathcal{P}(\hat{r}|a, z_i, q)$. This approach is related to the literature on confidence-weighted majority voting (Nitzan and Paroush, 1982; Grofman et al., 1983; Meyen et al., 2021), which posits that the theoretically optimal method for aggregating individual confidences is confidence-weighted majority voting, where more reliable individual responses are given greater weight.

3.1 Design Choices

Weighted Training. Reweighting has been a crucial technique in various machine learning domains, particularly for enhancing the robustness of training in the presence of significant class imbalances (Philip and Chan, 1998; Bankes et al., 2023; Huang et al., 2016; Cui et al., 2019). This technique is well-suited to our setting, as our data samples typically include a few successful instances alongside a large number of failure cases. Specifically, for each question, we generate multiple answers from the training set, assigning a weight of 1.0 to the negative samples. The weight of the positive samples is then set as the ratio of the number of negative samples to the number of positive samples. This approach ensures that the positive samples are appropriately emphasized, addressing the imbalance and improving overall model performance.

LLM Logits. We further study whether to incorporate the logit generated from the LLM alongside the LLM embedding. This transforms the learning problem into mapping the LLM embedding to the scaling factor necessary to adjust the LLM logit to match the ground truth reward. Pre-

vious literature has explored the extent to which the LLM-generated logit can accurately represent uncertainty (Malinin and Gales, 2020; Lin et al., 2023; Tian et al., 2023). Given token probabilities $\{p_1, ..., p_K\}$ from a token sequence of length K, we adopt two different ways to compute the sequence-level logit. The first is to compute the final logit as $\exp \sum_i \log(p_i), i = 1, ..., K$, which we denoted as Exp Sum Log (E.S.L.). As suggested by recent literature (Gupta et al., 2024), we also adopt a method where $\{p_1, ..., p_K\}$ are first sorted, and the second smallest value is then used to be the final sequence logit. We denote this as SORT.

Temperature Scaling. Temperature scaling is a widely used post-processing calibration method for neural networks (Guo et al., 2017). This technique is particularly effective for adjusting the confidence of predictions without altering the model's accuracy. Specifically, the calibration model produces logits z. The softmax function is applied to these logits, resulting in the original output probabilities $p = \text{softmax}(z)$. In temperature scaling, the logits z are divided by a temperature parameter $T > 0$ before applying the softmax function. This process modifies the distribution of the probabilities, yielding calibrated probabilities $q = \text{softmax}(z/T)$. The temperature parameter T is optimized using the negative log-likelihood (NLL) loss on the validation set. Note that while temperature scaling adjusts the confidence scores, the parameter T does not change the class with the highest probability. Therefore, the final accuracy of the model remains unaffected, but the output probabilities become better calibrated. By appropriately scaling the logits, the model's predicted probabilities become more aligned with the true likelihoods, improving the overall reliability of the predictions.

4 Experiments

4.1 Datasets

We evaluate our method on widely-used code-generation benchmarks, specifically MATH (Hendrycks et al., 2021) and TabMWP (Lu et al., 2022). The task statistics for our experiments are detailed in Table 4 in Appendix A. **MATH** (Hendrycks et al., 2021) is a benchmark comprising challenging competition mathematics problems. Each problem in MATH has a complete step-by-step solution and allows the use of code to obtain the final answer. **TabMWP** (Lu et al., 2022) is a math reasoning dataset featuring relatively straightforward questions such as numerical calculations based on relational tables. We employ 500 questions from the training set to train the calibration model and 500 questions from the test set for evaluating the results for TabMWP.

4.2 Language Agents

We study applications of our framework to both *static* and *dynamic* tool-using agents.

Static Tool-Using Agents. We first consider static tool-using language agents that utilize a fixed set of tools. We feed agents with the Primitive and Instance prompts in (Wang et al., 2024b). The Primitive instructs agents to generate programs using primitive functions, which is the standard approach for program-aided problem solving without tool induction (Cheng et al., 2022; Gao et al., 2023). The Instance prompt allows agents to compose high-level functions to solve tasks accordingly (Qian et al., 2023). We perform sampling with both of the prompts to collect multiple tool execution traces.

Dynamic Tool-Using Agents. In addition to static tool-using agents, our experiments extend to dynamic language agents capable of adaptively constructing and utilizing tools tailored to specific inputs. One notable example is the TroVE framework (Wang et al., 2024b). In TroVE, three distinct prompts are employed for each problem instance, offering choices to import pre-existing tools, create new tools, or opt for a simpler approach by relying solely on primitive functions. For every problem, each prompt is sampled identically K times, and the final solution is determined through self-consistency via a majority vote from the resulting pool of $3K$ answers. TroVE can deliver solutions that are simpler compared to static tool-using language agents.

4.3 Settings

Implementation Details. We primarily use `CODELLAMA-7B-INSTRUCT-hf` in our experiments without extra notice. The embedding dimension used in our experiments is 4096. We implement the calibration model as a multi-layer perceptron (MLP) consisting of three hidden layers with 256, 256, and 80 units, respectively. The classifier is trained for 10 epochs using Adam optimizer with a learning rate 1e-4, which can be completed within minutes given the training set. For model selection, we split the training dataset into training and validation sets with a ratio of 9:1. The final model is selected based on the minimum loss achieved on

Table 1: Experiment results for static and dynamic tool-using in TabMWP and Algebra. Our approach, despite being simple, outperforms baseline methods consistently and substantially. Specifically, PROBECAL-PROMPT&TRACE achieves the best performance in all settings. The ECE for INSTANCE, INSTANCE-SAMPLE, TROVE, and TROVE-SAMPLE are computed between all one vectors and the label, reflecting each prompt and trace is treated equally. The ECE for PROBECAL-PROMPT&TRACE is computed between the average of PROBECAL-PROMPT and PROBECAL-TRACE logits, and the label. Acc@k refers to the accuracy with the number of samples being k.

Method	**TabMWP**				**Algebra**			
	Acc@1	Acc@5	Acc@10	ECE	Acc@1	Acc@5	Acc@10	ECE
Static Tool-Using								
INSTANCE	30.90	48.54	51.79	0.690	17.69	26.45	28.96	0.823
INSTANCE-SAMPLE	34.49	50.09	52.70	0.690	18.26	26.92	29.30	0.823
INSTANCE-SORT	30.90	48.52	51.97	0.305	17.69	26.77	29.08	0.407
INSTANCE-E.S.L.	30.90	49.25	52.64	0.163	17.69	26.38	28.22	0.048
PROBECAL-PROMPT	**40.14**	51.29	53.87	0.067	**19.64**	27.68	29.94	0.052
PROBECAL-TRACE	30.90	51.97	56.35	0.057	17.69	27.98	30.68	0.047
PROBECAL-PROMPT&TRACE	**40.14**	**53.92**	**57.31**	**0.035**	**19.64**	**28.88**	**31.23**	**0.026**
Dynamic Tool-Using								
TROVE	21.27	36.69	42.00	0.788	15.06	28.47	32.03	0.851
TROVE-SAMPLE	23.19	38.51	43.59	0.788	16.45	29.04	32.10	0.851
TROVE-SORT	21.27	37.49	42.38	0.387	15.06	28.30	31.70	0.396
TROVE-E.S.L.	21.27	37.13	41.39	0.148	15.06	27.85	30.92	0.096
PROBECAL-PROMPT	**28.50**	42.15	46.16	0.078	**19.92**	30.17	32.78	0.054
PROBECAL-TRACE	21.27	41.54	48.47	0.072	15.06	29.26	32.86	0.051
PROBECAL-PROMPT&TRACE	**28.50**	**45.87**	**50.93**	**0.043**	**19.92**	**31.05**	**33.60**	**0.027**

the validation set. The temperature parameter is trained on the validation set when applying temperature scaling.

Metric. We use the task accuracy and Expected Calibration Error (ECE) (Guo et al., 2017) as the main metrics. ECE measures the calibration of a model's predicted probabilities by approximating the difference in expectation between confidence and accuracy. It is defined as $\sum_{m=1}^{M} \frac{B}{n} |\text{acc}(B_m) - \text{conf}(B_m)|$, where M is the number of bins and n is the number of samples, B_m is the set of samples whose predicted probabilities fall into the m-th bin, $\text{acc}(B_m)$ is the accuracy and $\text{conf}(B_m)$ is the average predicted confidence in m-th bin. We choose bin numbers to be 15 in the experiments.

Baselines. Our first baseline is denoted as INSTANCE in the static setting and TROVE in the dynamic setting, where each prompt is sampled uniformly and the final trace is selected based on self-consistency without confidence-weighting (Wang et al., 2022). We further compare with the baseline where each prompt is sampled proportional to the total number of positive answers generated by that prompt in the training set, with the final result selected using self-consistency without confidence-weighting. We denote it as INSTANCE-SAMPLE and TROVE-SAMPLE in the static and dynamic settings, respectively. We also compare with baselines where each prompt is selected uniformly, and the trace is selected based on a confidence-weighted majority vote using LLM logits, resulting in methods with suffixes E.S.L. and SORT.

4.4 Main Results

We begin by applying our embedding-based reward estimation method to prompt calibration (PROBECAL-PROMPT), trace calibration (PROBECAL-TRACE), and a combination of both prompt and trace calibration (PROBECAL-PROMPT&TRACE), and compare them against our baselines. To evaluate prompt selection calibration performance, we initially sample T answers from various prompts to create a subset of samples, with and without prompt calibration. Subsequently, we select the final answer from this pool of T samples, with (self-consistency with confidence-weighting) and

Table 2: Experimental results for Count, Geometry, TabMWP, and Algebra for static and dynamic tool-using settings with different variants.

Method	Count		Geometry		TabMWP		Algebra	
	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE
Static Tool-Using								
PROBECAL-TRACE	28.03	0.072	11.94	0.036	56.35	0.057	30.68	0.047
PROBECAL-TRACE-TS	28.05	0.079	11.97	0.034	56.40	0.060	30.68	0.051
PROBECAL-TRACE-WEIGHT	**28.06**	**0.066**	**12.03**	**0.029**	**56.52**	0.052	**30.75**	**0.046**
PROBECAL-TRACE-SORT	27.88	0.072	11.16	0.040	56.25	**0.050**	30.66	0.048
PROBECAL-TRACE-E.S.L.	26.84	0.118	10.39	0.203	53.84	0.144	29.84	0.118
Dynamic Tool-Using								
PROBECAL-TRACE	27.55	0.052	9.26	0.024	53.40	0.072	35.23	0.051
PROBECAL-TRACE-TS	27.54	0.055	9.26	0.025	53.50	0.079	35.21	0.055
PROBECAL-TRACE-WEIGHT	**27.58**	**0.049**	**9.27**	**0.019**	**53.72**	**0.071**	**35.24**	**0.049**
PROBECAL-TRACE-SORT	27.42	0.054	9.03	0.022	53.68	0.075	34.97	0.050
PROBECAL-TRACE-E.S.L.	26.67	0.093	6.72	0.127	51.98	0.083	34.44	0.100

without trace calibration (self-consistency without confidence-weighting). We vary T from 1 to 20 for dynamic and 1 to 10 for static tool-using cases. The final results are averaged over 5 runs. Within each run, performance is averaged across 10 instances of sampling T answers.

We present a subset of results in Table 1, focusing on tasks including TabMWP and Algebra in both static and dynamic tool-using contexts. Despite its simplicity, our method consistently outperforms baseline approaches in both settings. Notably, PROBECAL-PROMPT&TRACE achieves the highest performance across all datasets in all tool-using scenarios, surpassing the baseline by over 7% in the 5 and 10 sample settings for TabMWP in the dynamic tool-using settings. While E.S.L. can produce a relatively calibrated logit, it does not always lead to performance improvement when being used in trace selection. We empirically find the average of the logits from PROBECAL-PROMPT and PROBECAL-TRACE produce the most calibrated logits measured with ECE, which is shown as the ECE score for PROBECAL-PROMPT&TRACE. Full results for each dataset are provided in Appendix B.4 and Appendix for static and dynamic settings respectively. Full results on the ECE loss are provided in Table 12 and 13 in Appendix B and the calibration curves for PROBECAL and LLM logit are provided in Appendix D.

4.5 Ablation Studies

Design Choice Variants. We compare the performance of our method with different variants of design choices. As discussed in Section 3.1, we consider variants of PROBECAL including whether to apply temperature scaling, utilize weighted training, and incorporate the logit generated from the LLM alongside the LLM embedding as the input to PROBECAL, with suffixes being TS, WEIGHT, SORT and E.S.L. respectively. We summarize a subset of results of PROBECAL-TRACE variants on tasks including Count, Geometry, TabMWP, and Algebra in Table 2 and the full results can be found in Appendix B. As seen in the tables, TS does not impact the model accuracy and calibration results significantly, which is expected considering it only multiplies all the LLM logits with a single constant. WEIGHT, on the other hand, can generally improve the model marginally and achieve the most robust performance compared to other designs. For the SORT and E.S.L. models, their performance can fluctuate, and can occasionally degrade the model performance significantly.

Effect of Temperature Scaling on ECE. We further investigate the effect of temperature scaling by investigating the ECE before and after temperature scaling on both the training and testing datasets, as shown in Table 12 and 13 in Appendix B. We observe that the PROBECAL model without temperature scaling can already produce a well-calibrated logit with a low ECE. Furthermore, we note a no-

table distribution shift between the training and test sets, resulting in a higher ECE loss on the test set compared to the training set. Moreover, a low train ECE loss does not always correspond to a low test ECE loss. We find that applying temperature scaling to the proposed calibration model method does not significantly improve either the ECE loss or task accuracy. This observation may stem from the calibration model already being generally well-calibrated and the distribution shift between the train and test sets.

Different Base Language Models. We also evaluate on other representative language models, including `Mistral-7B` (Jiang et al., 2023), `CODELLAMA-13B-INSTRUCT-hf` and `Llama3-8b-Instruct`. Results of `Mistral-7B` in dynamic tool-using setting are summarized in Table 3. Further results can be found in Table 5 and 15 in the Appendix. We find that our proposed method can still deliver improvement when applied to a different base language model. For example, our method can improve the baseline accuracy by up to 5% on TabMWP as shown in Table 3, demonstrating the generality of our method to different LLM-based language agents.

5 Additional Experiment Results

5.1 Effect of the Size of the Training Set

To further analyze the generalization ability of our method, we conducted experiments with different sizes of training sets ranging from 50 to 500 questions for TabMWP. The results for `CODELLAMA-13B-INSTRUCT-hf` on TabMWP in the static tool-using setting are summarized in Table 14 in Appendix C. Our results show that our method PROBECAL can still deliver performance improvement even with only 50 questions from the training set.

5.2 Verbal Confidence

We further compare the proposed method with the method of verbal confidence (Tian et al., 2023), where we instructed the LLM to output the probability between 0 to 1 to represent the uncertainty itself. We summarize the results of using `CODELLAMA-13B-INSTRUCT-hf` on TabMWP in the static tool-using setting in Table 16 in Appendix C. We find that verbal confidence performs badly for the tool-using reasoning tasks we considered, and fails to bring improvement to the original baseline.

5.3 Closed-source LLMs

While we focus on the experimentes with open-source LLMs due to their transparency in this paper, we also conducted experiments with closed-source LLMs. For instance, We find that GPT-4o-mini (500 output length) results in an accuracy of 75.80%, but with a ECE of 0.5149 for E.S.L. and 0.1183 for SORT on Math Count & prob. This demonstrates that our claim "LLMs are not well-calibrated for tool-using scenarios" still holds for these models.

6 Related Work

Tool-Using Language Agents. Tools serve as external functional interfaces that significantly enhance and extend the task-solving capabilities of language models (Wang et al., 2024a). Recently, there has been a surge of interest in equipping language models with various tools, such as calculators, search engines, and general code programs. A notable example in this line of research is Toolformer (Schick et al., 2024), which integrates tools like Wikipedia search and machine translation systems. Toolformer utilizes GPT-J models (Wang and Komatsuzaki, 2021) as a baseline, fine-tuning the model on self-supervised, model-synthesized examples. Another significant development is ToolLLM (Qin et al., 2023), which fine-tunes LLaMA using instructions generated from ChatGPT. Additionally, ToolkenGPT (Hao et al., 2024) represents each tool as a token, learning their embeddings and augmenting the original vocabulary, thus eliminating the need for fine-tuning. TroVE (Wang et al., 2024b) prompts code language models to curate reusable high-level functions for writing solutions. However, addressing the uncertainty involved in using these tool-using agents has been rarely studied.

Uncertainty in Large Language Models. Various works have explored different methods to quantify the uncertainty within language models. These methods include the utilization of token-level logit (Malinin and Gales, 2020; Gupta et al., 2024), multiple response generation as a proxy (Lin et al., 2023), and enabling the language model itself to express confidence verbally (Tian et al., 2023). However, only a few of them study uncertainty calibrations for language or embodied agents(Ahn et al., 2022; Ren et al., 2023). Among them, Han et al. (Han et al., 2024) propose the UALA framework, which uses uncertainty as a metric to switch between the language model's own reasoning trajec-

Table 3: Results for Count and TabMWP in Dynamic Tool-Using setting using Mistral-7B. Our approach can still deliver improvement when applied to a different base language model. The full results can be found in Table 5. Acc@k refers to the accuracy with the number of samples being k.

Method	Count				TabMWP			
	Acc@1	Acc@5	Acc@20	ECE	Acc@1	Acc@5	Acc@20	ECE
TROVE	18.00	26.00	29.97	0.820	32.59	56.07	61.07	0.668
PROBECAL-PROMPT	**20.00**	26.98	**30.53**	0.059	**46.60**	60.53	63.56	0.087
PROBECAL-TRACE	18.00	26.54	29.58	0.062	32.59	59.06	65.77	0.084
PROBECAL-PROMPT&TRACE	**20.00**	**27.59**	29.87	**0.037**	**46.60**	**63.01**	**66.56**	**0.041**

tory and the use of external tools, highlighting the importance of incorporating uncertainty in the design of language model agents. While in this paper we focus on the task of tool-using where the output is completely operated by external tool execution, such as code generation, it would be an interesting direction to further integrate our calibration method with reasoning frameworks that utilize uncertainty such as UALA. Additionally, research has been conducted on investigating uncertainty calibration in few-shot learning (Zhao et al., 2021), long-form generation (Huang et al., 2024), and knowledge discovery settings (Jiang et al., 2021; Burns et al., 2022). Distinct from these works, our focus is on language agents where external tools are involved.

Probing. Probing is a growing area within machine learning interpretability (Elazar et al., 2021; Belinkov, 2022). Similar to our approach, probing in large language models involves training a small classification model, often linear, on the internal representations of the language model. This technique is used to uncover various types of information, including linguistic properties and significant behaviors of the LLM, such as truthfulness (Azaria and Mitchell, 2023; Li et al., 2024b,a). Different from this line of work, our focus is to align the confidence level of language agents with their actual success rates instead of interpreting the model behaviors.

7 Conclusion

This paper addresses the inherent uncertainties in tool-using language agents, crucial for performing complex tasks. Despite advancements, challenges persist due to variability and errors from integrating external tools. Our framework recalibrates probability estimates for tool-using decisions to improve robustness and effectiveness. We focused on two primary areas of miscalibration: language agent prompts and tool execution traces. We systematically addressed these uncertainties by training a classification model using embeddings from a large language model and employing complementary methods like reweighting during training and temperature scaling during inference. Experiments on the MATH and TabMWP datasets demonstrate improvements in calibration and performance, underscoring the effectiveness of our approach. Our work fills a critical research gap, enhancing the reliability and accuracy of tool-using language agents.

Acknowledgement

YW is supported in part by the Office of Naval Research under grant number N00014-23-1-2590, the National Science Foundation under Grant No. 2231174, No. 2310831, No. 2428059, and a Michigan Institute for Data Science Propelling Original Data Science (PODS) grant.

Limitations

It is worth noting that our framework requires training an external model for calibration, which introduces computational overhead. The proposed framework has the potential to yield significant positive societal impacts by enhancing the reliability and effectiveness of tool-using language agents. However, providing agents with ill-intentioned tools could lead to negative consequences. Therefore, careful guidance on tool use should be provided before deploying these models to ensure ethical and responsible application.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Amos Azaria and Tom Mitchell. 2023. The internal state of an llm knows when its lying. *arXiv preprint arXiv:2304.13734*.

William Bankes, George Hughes, Ilija Bogunovic, and Zi Wang. 2023. Reducr: Robust data downsampling using class priority reweighting. In *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2022. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*.

Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. 2017. Boltzmann exploration done right. *Advances in neural information processing systems*, 30.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. 2022. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175.

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.

Bernard Grofman, Guillermo Owen, and Scott L Feld. 1983. Thirteen theorems in search of the truth. *Theory and decision*, 15(3):261–278.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. 2024. Language model cascades: Token-level uncertainty and beyond. *arXiv preprint arXiv:2404.10136*.

Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. 2024. Towards uncertainty-aware language agent. *arXiv preprint arXiv:2401.14016*.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2024. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems*, 36.

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*.

Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. 2016. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5375–5384.

Yukun Huang, Yixin Liu, Raghuveer Thirukovalluru, Arman Cohan, and Bhuwan Dhingra. 2024. Calibrating long-form generations from large language models. *arXiv preprint arXiv:2402.06544*.

Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709.

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*.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Karim Lasri, Tiago Pimentel, Alessandro Lenci, Thierry Poibeau, and Ryan Cotterell. 2022. Probing for the usage of grammatical number. *arXiv preprint arXiv:2204.08831*.

Kenneth Li, Samy Jelassi, Hugh Zhang, Sham Kakade, Martin Wattenberg, and David Brandfonbrener. 2024a. Q-probe: A lightweight approach to reward maximization for language models. *arXiv preprint arXiv:2402.14688*.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024b. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2023. Generating with confidence: Uncertainty quantification for black-box large language models. *arXiv preprint arXiv:2305.19187*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2024. Chameleon: Plug-and-play compositional reasoning with large language models. *Advances in Neural Information Processing Systems*, 36.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2022. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. *arXiv preprint arXiv:2209.14610*.

Andrey Malinin and Mark Gales. 2020. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*.

Sascha Meyen, Dorothee MB Sigg, Ulrike von Luxburg, and Volker H Franz. 2021. Group decisions based on confidence weighted majority voting. *Cognitive research: principles and implications*, 6:1–13.

Shmuel Nitzan and Jacob Paroush. 1982. Optimal decision rules in uncertain dichotomous choice situations. *International Economic Review*, pages 289–297.

K Philip and SJS Chan. 1998. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 164–168. AAAI Press Manchester, UK.

Cheng Qian, Chi Han, Yi R Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. 2023. Creator: Disentangling abstract and concrete reasonings of large language models through tool creation. *arXiv preprint arXiv:2305.14318*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Allen Z Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, et al. 2023. Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36.

Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11888–11898.

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*.

Ben Wang and Aran Komatsuzaki. 2021. Gpt-j-6b: A 6 billion parameter autoregressive language model.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zhiruo Wang, Zhoujun Cheng, Hao Zhu, Daniel Fried, and Graham Neubig. 2024a. What are tools anyway? a survey from the language model perspective. *arXiv preprint arXiv:2403.15452*.

Zhiruo Wang, Daniel Fried, and Graham Neubig. 2024b. Trove: Inducing verifiable and efficient toolboxes for solving programmatic tasks. *arXiv preprint arXiv:2401.12869*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

A Task statistics

We show the statistics of the dataset used in our experiments in Table.

Table 4: Task Statistics

Task	Dataset	# of train	# of test
Math	Precalculus	198	156
	Geometry	474	237
	Count & prob.	483	291
	Number	780	497
	Algebra	1065	881
	Prealgebra	814	636
	Intermediate	607	503
TableQA	TabMWP	500	500

B Experimental Results

B.1 Different base language model

We here show the full results for Count and TabMWP in the Dynamic Tool-Using setting using Mistral-7B in Table 5. Our approach can still deliver improvement when applied to a different base language model.

B.2 Static Tool-Using results

We here show the full results of Static Tool-Using settings on each dataset in Table 6.

B.3 Static Tool-Using results of different variants

We here show the full results of Static Tool-Using settings of different variants on each dataset in Table 7 and 8.

B.4 Dynamic Tool-Using results on each dataset

We here show the full experimental results on each dataset in the dynamic tool-using setting in Table 9.

B.5 Dynamic Tool-Using results of different variants on each dataset

We here show the full experimental results of different variants on each dataset in the dynamic tool-using setting in Table 10 and 11.

B.6 Static Tool-Using ECE results

We here show the full experimental results of ECE loss on each dataset in the static tool-using settings in Table 12

B.7 Dynamic Tool-Using ECE results

We here show the full experimental results of ECE loss on each dataset in the static tool-using settings in Table 13.

C More Experimental Results

We here show additional experimental results on the effect of the size of the training set, the performance of other language models and verbal confidence in Table 14, 15 and 16.

D Calibration Curve

We show examples of the calibration curve of PROBECAL and the LLM logits for each dataset in both static and dynamic settings. The calibration curve for PROBECAL-PROMPT&TRACE is computed between the average of the PROBECAL-PROMPT and PROBECAL-TRACE logits, and the label.

In each figure of calibration curve, the X-axis represents the predicted probability (model confidence), the Y axis represents the accuracy, the red line (y=x) demonstrates perfect calibration, the blue curve is the model's calibration curve, and the grey histogram shows the number of instances within each confidence bin.

D.1 Static tool-using calibration curve

We here show the calibration curve of PROBECAL and LLM logit including SORT and E.S.L. for each dataset in the static tool-using setting.

D.2 Dynamic tool-using calibration curve

We here show the calibration curve of PROBECAL and LLM logit including SORT and E.S.L. for each dataset in the dynamic tool-using setting.

Table 5: Results for Count and TabMWP in Dynamic Tool-Using setting using Mistral-7B. Our approach can still deliver improvement when applied to a different base language model.

Method	**Count**							
#Samples	1	2	3	4	5	10	15	20
TROVE	18.00	21.42	23.68	25.08	26.00	28.47	29.51	29.97
PROBECAL-PROMPT	20.00	23.12	24.65	26.25	26.98	29.12	30.23	30.53
PROBECAL-TRACE	18.00	22.42	24.54	25.84	26.54	28.41	29.11	29.58
PROBECAL-PROMPT&TRACE	20.00	24.16	25.77	26.97	27.59	28.99	29.50	29.87
Method	**TabMWP**							
#Samples	1	2	3	4	5	10	15	20
TROVE	32.59	44.79	50.75	53.84	56.07	59.80	60.76	61.07
PROBECAL-PROMPT	46.60	54.87	57.86	59.66	60.53	62.38	63.13	63.56
PROBECAL-TRACE	32.59	46.22	52.72	56.54	59.06	63.66	65.21	65.77
PROBECAL-PROMPT&TRACE	46.60	56.08	59.72	61.93	63.01	65.37	66.23	66.56

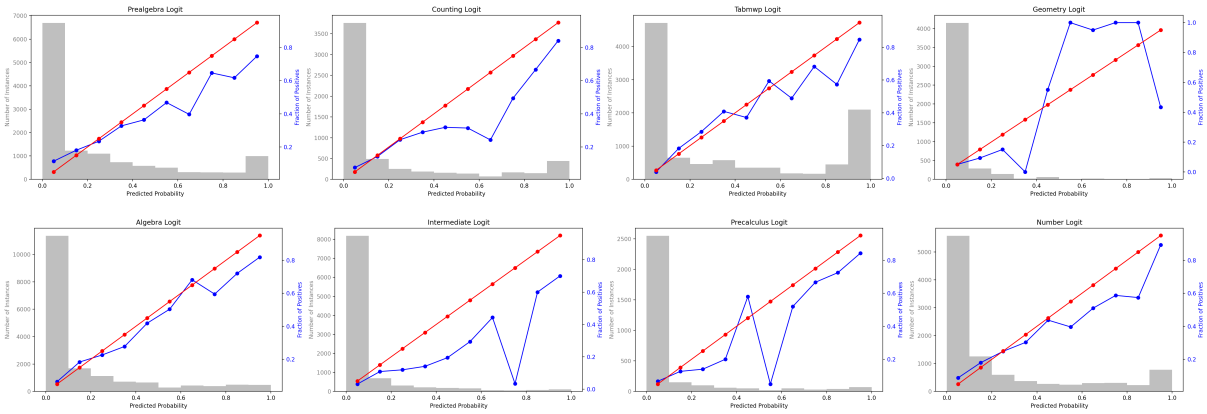

Figure 3: Calibration curve of PROBECAL-PROMPT logits of LLM in Static Tool-Using

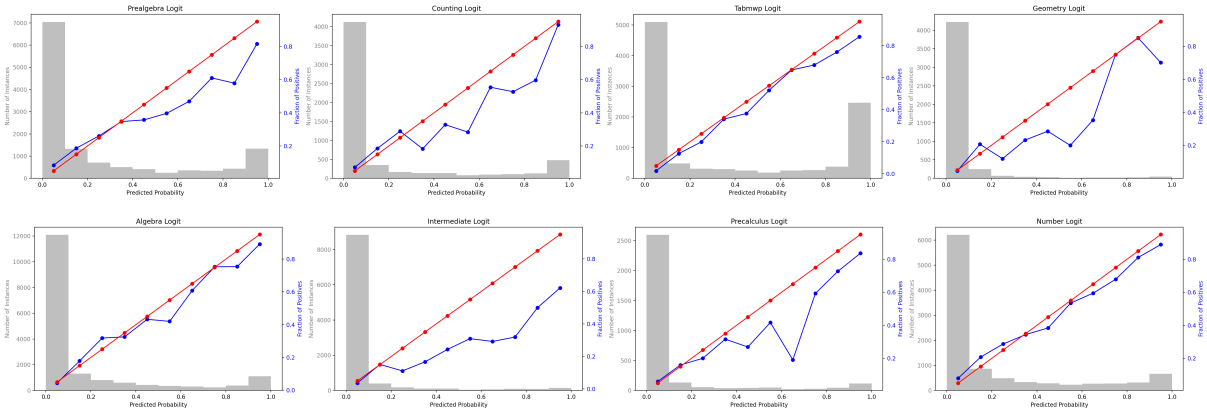

Figure 4: Calibration curve of PROBECAL-TRACE logits of LLM in Static Tool-Using

Table 6: Static Tool-Using results on each dataset

Dataset	**Prealgebra**							
# Samples	1	2	3	4	5	7	9	10
INSTANCE	25.02	28.32	30.01	31.20	31.87	32.62	33.00	33.21
INSTANCE-SAMPLE	25.30	28.25	30.19	31.29	31.91	32.53	32.92	33.24
INSTANCE-SORT	25.02	28.47	30.17	31.29	32.13	32.53	32.90	33.08
INSTANCE-E.S.L.	25.02	28.76	29.85	30.87	31.28	31.63	31.86	32.09
PROBECAL-PROMPT	25.99	28.84	30.54	31.63	32.09	33.06	33.43	33.69
PROBECAL-TRACE	25.02	29.05	30.91	31.99	32.60	33.35	33.75	33.96
PROBECAL-PROMPT&TRACE	25.99	29.74	31.24	32.21	32.74	33.47	33.92	34.01
Dataset	**Count**							
# Samples	1	2	3	4	5	7	9	10
INSTANCE	19.60	21.30	23.68	24.47	25.17	26.09	26.61	27.06
INSTANCE-SAMPLE	20.18	21.59	23.93	24.72	25.43	26.20	26.83	26.72
INSTANCE-SORT	19.60	22.62	24.57	25.06	25.52	26.13	26.55	26.92
INSTANCE-E.S.L.	19.60	23.60	24.90	25.66	25.79	26.39	26.52	26.92
PROBECAL-PROMPT	22.44	23.87	26.07	26.96	27.50	28.18	28.62	29.01
PROBECAL-TRACE	19.60	23.97	25.62	26.41	26.82	27.47	27.89	28.03
PROBECAL-PROMPT&TRACE	22.44	25.70	26.70	27.48	27.58	28.08	28.44	28.69
Dataset	**TabMWP**							
# Samples	1	2	3	4	5	7	9	10
INSTANCE	30.90	40.68	44.72	47.11	48.54	50.54	51.27	51.79
INSTANCE-SAMPLE	34.49	43.58	46.99	48.75	50.09	51.42	52.32	52.70
INSTANCE-SORT	30.90	40.40	44.54	46.92	48.52	50.54	51.49	51.97
INSTANCE-E.S.L.	30.90	40.60	45.16	47.73	49.25	51.22	52.12	52.64
PROBECAL-PROMPT	40.14	46.24	48.87	50.20	51.29	52.60	53.34	53.87
PROBECAL-TRACE	30.90	41.34	46.72	49.96	51.97	54.57	55.81	56.35
PROBECAL-PROMPT&TRACE	40.14	47.43	50.67	52.47	53.92	55.83	56.79	57.31
Dataset	**Geometry**							
# Samples	1	2	3	4	5	7	9	10
INSTANCE	7.35	8.48	9.54	10.10	10.77	11.34	11.70	11.95
INSTANCE-SAMPLE	7.49	8.78	9.65	10.17	10.78	11.35	11.81	11.88
INSTANCE-SORT	7.35	8.85	9.93	10.44	10.88	11.39	11.61	11.90
INSTANCE-E.S.L.	7.35	8.82	9.66	10.25	10.42	10.56	10.61	10.81
PROBECAL-PROMPT	8.28	9.51	10.16	10.86	11.12	11.64	11.97	11.90
PROBECAL-TRACE	7.35	9.13	10.04	10.67	11.10	11.64	11.81	11.94
PROBECAL-PROMPT&TRACE	8.28	9.95	10.53	11.11	11.32	11.64	11.72	11.89
Dataset	**Algebra**							
# Samples	1	2	3	4	5	7	9	10
INSTANCE	17.69	21.83	24.21	25.70	26.45	27.77	28.75	28.96
INSTANCE-SAMPLE	18.26	22.32	24.48	25.92	26.92	28.12	28.87	29.30
INSTANCE-SORT	17.69	21.97	24.49	25.97	26.77	27.97	28.85	29.08
INSTANCE-E.S.L.	17.69	22.03	24.38	25.70	26.38	27.38	28.13	28.22
PROBECAL-PROMPT	19.64	23.47	25.61	26.85	27.68	28.86	29.67	29.94
PROBECAL-TRACE	17.69	22.74	25.47	27.08	27.98	29.44	30.43	30.68
PROBECAL-PROMPT&TRACE	19.64	24.49	26.75	28.06	28.88	30.14	30.99	31.23
Dataset	**Intermediate**							
# Samples	1	2	3	4	5	7	9	10
INSTANCE	6.39	9.71	11.29	12.47	13.20	14.12	14.74	15.03
INSTANCE-SAMPLE	6.40	9.63	11.30	12.39	13.08	14.00	14.68	14.91
INSTANCE-SORT	6.39	9.80	11.45	12.64	13.33	14.28	14.86	15.22
INSTANCE-E.S.L.	6.39	9.82	11.50	12.79	13.56	14.39	14.97	15.33
PROBECAL-PROMPT	7.82	10.39	11.69	12.65	13.25	14.19	14.81	14.96
PROBECAL-TRACE	6.39	9.84	11.51	12.80	13.62	14.58	15.32	15.63
PROBECAL-PROMPT&TRACE	7.82	10.48	11.84	12.91	13.59	14.43	15.12	15.35
Dataset	**Precalculus**							
# Samples	1	2	3	4	5	7	9	10
INSTANCE	11.61	15.42	17.86	19.40	20.09	21.53	22.32	22.59
INSTANCE-SAMPLE	12.91	16.92	19.08	20.04	21.27	22.33	23.21	23.39
INSTANCE-SORT	11.61	15.75	18.43	20.13	20.93	22.07	22.64	22.82
INSTANCE-E.S.L.	11.61	16.18	18.44	20.20	20.60	21.89	22.30	22.56
PROBECAL-PROMPT	14.09	17.44	19.43	20.36	21.18	22.35	23.22	23.51
PROBECAL-TRACE	11.61	16.18	18.50	20.08	20.65	21.91	22.65	22.69
PROBECAL-PROMPT&TRACE	14.09	17.87	19.63	20.56	21.27	22.10	22.72	22.85
Dataset	**Number**							
# Samples	1	2	3	4	5	7	9	10
INSTANCE	23.84	25.15	27.53	28.38	29.36	30.50	31.45	31.68
INSTANCE-SAMPLE	24.25	25.64	27.97	29.17	30.01	31.03	31.98	31.97
INSTANCE-SORT	23.84	26.71	28.65	29.28	30.03	30.94	31.66	31.91
INSTANCE-E.S.L.	23.84	26.78	28.42	29.03	29.66	30.50	31.08	31.23
PROBECAL-PROMPT	25.02	26.50	28.50	29.34	30.29	31.16	31.69	31.87
PROBECAL-TRACE	23.84	27.07	28.99	31.04	31.94	32.90	33.48	33.72
PROBECAL-PROMPT&TRACE	25.02	28.71	30.46	31.33	32.11	33.03	33.58	33.60

Table 7: Static Tool-Using results of different variants on each dataset - I

Dataset	**Prealgebra**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	25.99	28.84	30.54	31.63	32.09	33.06	33.43	33.69
ProbeCal-Trace	25.02	29.05	30.91	31.99	32.60	33.35	33.75	33.96
ProbeCal-Prompt&Trace	25.99	29.74	31.24	32.21	32.74	33.47	33.92	34.01
ProbeCal-Prompt-TS	25.90	28.97	30.60	31.52	32.10	33.00	33.38	33.59
ProbeCal-Trace-TS	25.02	29.05	30.91	31.98	32.58	33.32	33.73	33.95
ProbeCal-Prompt&Trace-TS	25.90	29.76	31.35	32.14	32.66	33.38	33.85	33.96
ProbeCal-Prompt-Weight	26.15	29.01	30.58	31.74	32.28	33.11	33.60	33.69
ProbeCal-Trace-Weight	25.02	29.04	30.90	31.95	32.65	33.35	33.69	33.99
ProbeCal-Prompt&Trace-Weight	26.15	29.91	31.29	32.18	32.69	33.41	33.84	33.93
ProbeCal-Prompt-Sort	26.13	28.86	30.53	31.52	32.02	32.82	33.36	33.48
ProbeCal-Trace-Sort	25.02	29.03	30.85	31.94	32.59	33.32	33.74	34.04
ProbeCal-Prompt&Trace-Sort	26.13	29.83	31.25	32.05	32.51	33.13	33.80	33.80
ProbeCal-Prompt-E.S.L.	25.70	28.51	30.03	30.99	31.72	32.52	33.02	33.29
ProbeCal-Trace-E.S.L.	25.02	28.74	30.42	31.42	32.10	33.02	33.37	33.72
ProbeCal-Prompt&Trace-E.S.L.	25.70	28.89	30.44	31.16	31.91	32.88	33.48	33.61
Dataset	**Count**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	22.44	23.87	26.07	26.96	27.50	28.18	28.62	29.01
ProbeCal-Trace	19.60	23.97	25.62	26.41	26.82	27.47	27.89	28.03
ProbeCal-Prompt&Trace	22.44	25.70	26.70	27.48	27.58	28.08	28.44	28.69
ProbeCal-Prompt-TS	22.48	23.84	25.92	26.84	27.59	28.25	28.81	28.75
ProbeCal-Trace-TS	19.60	23.97	25.60	26.40	26.79	27.47	27.87	28.05
ProbeCal-Prompt&Trace-TS	22.48	25.91	26.65	27.32	27.60	27.98	28.27	28.49
ProbeCal-Prompt-Weight	22.50	23.84	25.88	26.64	27.45	28.08	28.60	28.77
ProbeCal-Trace-Weight	19.60	24.00	25.68	26.42	26.90	27.71	27.91	28.06
ProbeCal-Prompt&Trace-Weight	22.50	25.79	26.68	27.23	27.72	28.19	28.51	28.56
ProbeCal-Prompt-Sort	22.51	23.98	25.98	26.53	27.31	27.81	28.38	28.46
ProbeCal-Trace-Sort	19.60	23.86	25.52	26.26	26.71	27.33	27.70	27.88
ProbeCal-Prompt&Trace-Sort	22.51	25.62	26.52	26.98	27.45	27.76	28.25	28.25
ProbeCal-Prompt-E.S.L.	21.62	22.71	25.18	25.51	26.53	27.24	27.66	27.98
ProbeCal-Trace-E.S.L.	19.60	23.05	24.48	25.13	25.50	26.11	26.74	26.84
ProbeCal-Prompt&Trace-E.S.L.	21.62	24.09	25.16	25.59	25.81	26.50	27.10	27.15
Dataset	**TabMWP**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	40.14	46.24	48.87	50.20	51.29	52.60	53.34	53.87
ProbeCal-Trace	30.90	41.34	46.72	49.96	51.97	54.57	55.81	56.35
ProbeCal-Prompt&Trace	40.14	47.43	50.67	52.47	53.92	55.83	56.79	57.31
ProbeCal-Prompt-TS	39.76	46.26	48.59	50.38	51.17	52.48	53.37	53.72
ProbeCal-Trace-TS	30.90	41.34	46.73	49.97	51.99	54.60	55.87	56.40
ProbeCal-Prompt&Trace-TS	39.76	47.51	50.43	52.71	53.78	55.55	56.78	57.37
ProbeCal-Prompt-Weight	40.35	46.26	48.86	50.35	51.31	52.66	53.51	53.84
ProbeCal-Trace-Weight	30.90	41.35	46.76	49.99	52.01	54.59	55.94	56.52
ProbeCal-Prompt&Trace-Weight	40.35	47.48	50.71	52.71	53.98	55.93	57.10	57.65
ProbeCal-Prompt-Sort	39.57	45.85	48.41	49.92	50.89	52.23	53.09	53.41
ProbeCal-Trace-Sort	30.90	41.40	46.67	49.91	51.89	54.44	55.67	56.25
ProbeCal-Prompt&Trace-Sort	39.57	47.03	50.04	51.99	53.30	55.17	56.28	56.81
ProbeCal-Prompt-E.S.L.	36.31	43.40	46.03	48.05	48.99	50.57	51.56	51.93
ProbeCal-Trace-E.S.L.	30.90	41.06	45.69	48.31	50.06	52.31	53.30	53.84
ProbeCal-Prompt&Trace-E.S.L.	36.31	43.88	46.73	48.88	49.99	51.72	52.77	53.06
Dataset	**Geometry**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	8.28	9.51	10.16	10.86	11.12	11.64	11.97	11.90
ProbeCal-Trace	7.35	9.13	10.04	10.67	11.10	11.64	11.81	11.94
ProbeCal-Prompt&Trace	8.28	9.95	10.53	11.11	11.32	11.64	11.72	11.89
ProbeCal-Prompt-TS	8.37	9.31	10.33	10.73	11.06	11.51	11.94	11.95
ProbeCal-Trace-TS	7.35	9.13	10.03	10.67	11.09	11.64	11.82	11.97
ProbeCal-Prompt&Trace-TS	8.37	9.79	10.70	11.08	11.34	11.53	11.95	11.94
ProbeCal-Prompt-Weight	8.42	9.33	10.48	10.90	11.30	11.67	12.03	11.91
ProbeCal-Trace-Weight	7.35	9.14	10.14	10.76	11.11	11.71	11.88	12.03
ProbeCal-Prompt&Trace-Weight	8.42	9.82	10.65	10.92	11.41	11.52	11.86	11.75
ProbeCal-Prompt-Sort	7.66	8.81	9.82	10.11	10.71	10.94	11.22	11.37
ProbeCal-Trace-Sort	7.35	8.93	9.74	10.19	10.46	10.79	11.00	11.16
ProbeCal-Prompt&Trace-Sort	7.66	8.93	9.63	9.71	9.98	10.33	10.43	10.51
ProbeCal-Prompt-E.S.L.	6.88	7.83	8.75	8.96	9.51	9.95	10.35	10.52
ProbeCal-Trace-E.S.L.	7.35	8.50	9.07	9.51	9.80	10.02	10.32	10.39
ProbeCal-Prompt&Trace-E.S.L.	6.88	7.93	8.46	8.84	9.22	9.61	9.83	10.01

Table 8: Static Tool-Using results of different variants on each dataset - II

Dataset	**Algebra**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	19.64	23.47	25.61	26.85	27.68	28.86	29.67	29.94
ProbeCal-Trace	17.69	22.74	25.47	27.08	27.98	29.44	30.43	30.68
ProbeCal-Prompt&Trace	19.64	24.49	26.75	28.06	28.88	30.14	30.99	31.23
ProbeCal-Prompt-TS	19.81	23.32	25.58	26.79	27.73	28.83	29.62	29.88
ProbeCal-Trace-TS	17.69	22.74	25.48	27.08	27.97	29.44	30.43	30.68
ProbeCal-Prompt&Trace-TS	19.81	24.31	26.67	27.96	28.98	30.17	30.98	31.22
ProbeCal-Prompt-Weight	19.90	23.53	25.86	27.04	27.77	29.03	29.65	29.94
ProbeCal-Trace-Weight	17.69	22.77	25.44	27.12	27.98	29.42	30.41	30.75
ProbeCal-Prompt&Trace-Weight	19.90	24.56	26.91	28.21	29.10	30.34	31.10	31.41
ProbeCal-Prompt-Sort	19.69	23.51	25.62	26.77	27.64	28.83	29.60	29.95
ProbeCal-Trace-Sort	17.69	22.74	25.43	27.02	27.94	29.36	30.30	30.66
ProbeCal-Prompt&Trace-Sort	19.69	24.41	26.64	27.94	28.85	30.02	30.88	31.28
ProbeCal-Prompt-E.S.L.	19.18	22.65	24.71	26.03	27.01	28.09	28.88	29.21
ProbeCal-Trace-E.S.L.	17.69	22.49	25.08	26.54	27.27	28.67	29.53	29.84
ProbeCal-Prompt&Trace-E.S.L.	19.18	23.40	25.34	26.60	27.57	28.59	29.48	29.83
Dataset	**Intermediate**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	7.82	10.39	11.69	12.65	13.25	14.19	14.81	14.96
ProbeCal-Trace	6.39	9.84	11.51	12.80	13.62	14.58	15.32	15.63
ProbeCal-Prompt&Trace	7.82	10.48	11.84	12.91	13.59	14.43	15.12	15.35
ProbeCal-Prompt-TS	7.78	10.33	11.83	12.64	13.21	14.10	14.83	15.05
ProbeCal-Trace-TS	6.39	9.84	11.51	12.80	13.62	14.59	15.32	15.64
ProbeCal-Prompt&Trace-TS	7.78	10.47	11.99	12.86	13.54	14.48	15.20	15.43
ProbeCal-Prompt-Weight	7.77	10.25	11.88	12.69	13.44	14.18	14.77	14.97
ProbeCal-Trace-Weight	6.39	9.83	11.54	12.80	13.64	14.59	15.37	15.71
ProbeCal-Prompt&Trace-Weight	7.77	10.41	12.11	12.97	13.72	14.59	15.17	15.36
ProbeCal-Prompt-Sort	7.58	10.20	11.37	12.34	12.97	13.77	14.54	14.71
ProbeCal-Trace-Sort	6.39	9.75	11.40	12.69	13.46	14.43	15.13	15.42
ProbeCal-Prompt&Trace-Sort	7.58	10.25	11.54	12.57	13.22	14.04	14.88	15.09
ProbeCal-Prompt-E.S.L.	6.64	8.90	10.24	11.03	11.64	12.55	13.09	13.22
ProbeCal-Trace-E.S.L.	6.39	9.63	11.19	12.34	13.07	13.91	14.54	14.85
ProbeCal-Prompt&Trace-E.S.L.	6.64	8.86	10.12	10.98	11.61	12.44	13.09	13.20
Dataset	**Precalculus**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	14.09	17.44	19.43	20.36	21.18	22.35	23.22	23.51
ProbeCal-Trace	11.61	16.18	18.50	20.08	20.65	21.91	22.65	22.69
ProbeCal-Prompt&Trace	14.09	17.87	19.63	20.56	21.27	22.10	22.72	22.85
ProbeCal-Prompt-TS	14.11	17.27	19.42	20.28	21.29	22.55	23.15	23.53
ProbeCal-Trace-TS	11.61	16.18	18.50	20.11	20.67	21.93	22.69	22.70
ProbeCal-Prompt&Trace-TS	14.11	17.86	19.64	20.56	21.34	22.30	22.88	22.95
ProbeCal-Prompt-Weight	14.25	17.74	19.74	20.83	21.45	22.39	23.28	23.47
ProbeCal-Trace-Weight	11.61	16.16	18.43	20.09	20.74	21.89	22.65	22.68
ProbeCal-Prompt&Trace-Weight	14.25	18.14	19.87	20.92	21.38	22.15	22.81	23.08
ProbeCal-Prompt-Sort	13.15	16.98	19.15	19.91	21.11	22.08	22.81	23.05
ProbeCal-Trace-Sort	11.61	16.17	18.45	20.07	20.59	21.82	22.53	22.60
ProbeCal-Prompt&Trace-Sort	13.15	17.45	19.36	19.91	20.79	21.34	21.76	21.97
ProbeCal-Prompt-E.S.L.	12.96	15.96	17.99	19.20	20.21	21.26	22.19	22.25
ProbeCal-Trace-E.S.L.	11.61	16.13	18.33	19.87	20.36	21.32	21.88	21.90
ProbeCal-Prompt&Trace-E.S.L.	12.96	16.34	17.91	19.07	19.55	20.22	20.97	21.00
Dataset	**Number**							
# Samples	1	2	3	4	5	7	9	10
ProbeCal-Prompt	25.02	26.50	28.50	29.34	30.29	31.16	31.69	31.87
ProbeCal-Trace	23.84	27.96	29.99	31.04	31.94	32.90	33.48	33.72
ProbeCal-Prompt&Trace	25.02	28.71	30.46	31.33	32.11	33.03	33.58	33.60
ProbeCal-Prompt-TS	25.07	26.31	28.36	29.28	30.05	31.10	31.75	31.86
ProbeCal-Trace-TS	23.84	27.96	29.99	31.04	31.95	32.94	33.52	33.77
ProbeCal-Prompt&Trace-TS	25.07	28.57	30.27	31.20	31.93	33.01	33.64	33.89
ProbeCal-Prompt-Weight	25.10	26.31	28.64	29.38	30.34	31.32	31.85	31.96
ProbeCal-Trace-Weight	23.84	27.91	29.96	30.90	31.73	32.74	33.47	33.75
ProbeCal-Prompt&Trace-Weight	25.10	28.51	30.43	31.26	32.05	33.00	33.56	33.75
ProbeCal-Prompt-Sort	25.25	26.72	28.69	29.49	30.24	31.18	31.72	31.87
ProbeCal-Trace-Sort	23.84	27.85	29.80	30.85	31.69	32.63	33.34	33.59
ProbeCal-Prompt&Trace-Sort	25.25	28.74	30.30	31.29	32.00	32.85	33.49	33.69
ProbeCal-Prompt-E.S.L.	24.64	26.22	28.22	29.14	29.84	30.67	31.43	31.53
ProbeCal-Trace-E.S.L.	23.84	27.33	29.13	29.94	30.68	31.62	32.42	32.57
ProbeCal-Prompt&Trace-E.S.L.	24.64	27.60	29.15	30.12	30.57	31.45	32.12	32.27

Table 9: Dynamic Tool-Using results on each dataset

Dataset	**Prealgebra**							
# Samples	1	2	3	4	5	10	15	20
TROVE	19.90	24.43	26.88	28.42	29.64	32.12	32.89	33.57
TROVE-SAMPLE	20.35	24.51	27.25	28.76	29.84	32.00	32.94	33.39
TROVE-SORT	19.90	24.90	27.28	28.78	29.78	31.93	32.78	33.36
TROVE-E.S.L.	19.90	25.05	27.12	28.38	29.14	30.95	31.76	32.17
PROBECAL-PROMPT	19.66	23.79	26.63	28.31	29.34	31.79	32.81	33.36
PROBECAL-TRACE	19.90	25.79	28.25	29.68	30.59	33.01	33.98	34.45
PROBECAL-PROMPT&TRACE	19.66	25.04	27.80	29.27	30.23	32.52	33.58	34.25
Dataset	**Count**							
# Samples	1	2	3	4	5	10	15	20
TROVE	16.97	20.59	21.96	23.05	23.68	25.45	26.18	26.49
TROVE-SAMPLE	18.41	21.50	22.91	23.71	24.34	25.83	26.24	26.54
TROVE-SORT	16.97	20.41	22.07	22.74	23.37	25.01	25.54	25.76
TROVE-E.S.L.	16.97	20.15	21.58	22.03	22.36	23.58	23.79	23.85
PROBECAL-PROMPT	20.14	22.79	24.17	24.98	25.61	26.59	27.11	27.28
PROBECAL-TRACE	16.97	21.29	23.45	24.34	25.16	26.59	27.32	27.55
PROBECAL-PROMPT&TRACE	20.14	23.56	24.78	25.52	26.13	27.06	27.63	27.76
Dataset	**TabMWP**							
# Samples	1	2	3	4	5	10	15	20
TROVE	21.27	28.16	32.34	34.84	36.69	42.00	44.00	45.56
TROVE-SAMPLE	23.19	30.01	33.93	36.76	38.51	43.59	45.69	46.84
TROVE-SORT	21.27	29.05	33.24	35.73	37.49	42.38	44.37	45.91
TROVE-E.S.L.	21.27	29.15	33.30	35.60	37.13	41.39	43.03	44.09
PROBECAL-PROMPT	28.50	34.60	38.08	40.49	42.15	46.16	48.13	48.87
PROBECAL-TRACE	21.27	30.23	35.58	39.06	41.54	48.47	51.45	53.40
PROBECAL-PROMPT&TRACE	28.50	36.82	40.93	43.83	45.87	50.93	53.32	54.34
Dataset	**Geometry**							
# Samples	1	2	3	4	5	10	15	20
TROVE	3.99	5.32	6.03	6.68	7.17	8.65	8.91	9.39
TROVE-SAMPLE	3.98	5.65	6.32	6.87	7.30	8.53	9.25	9.69
TROVE-SORT	3.99	5.36	6.31	6.78	7.28	8.55	8.92	9.32
TROVE-E.S.L.	3.99	5.43	6.32	6.73	7.16	7.93	7.84	8.21
PROBECAL-PROMPT	4.67	5.99	6.70	7.32	7.66	8.88	9.58	10.01
PROBECAL-TRACE	3.99	5.44	6.33	6.89	7.40	8.41	8.80	9.26
PROBECAL-PROMPT&TRACE	4.67	6.23	6.86	7.56	7.76	8.70	9.28	9.61
Dataset	**Algebra**							
# Samples	1	2	3	4	5	10	15	20
TROVE	15.06	21.12	24.64	26.89	28.47	32.03	33.39	34.14
TROVE-SAMPLE	16.45	22.66	25.61	27.82	29.04	32.10	33.36	33.96
TROVE-SORT	15.06	21.14	24.58	26.79	28.30	31.70	33.14	33.90
TROVE-E.S.L.	15.06	21.12	24.47	26.50	27.85	30.92	32.24	32.82
PROBECAL-PROMPT	19.92	25.13	27.66	29.13	30.17	32.78	33.83	34.42
PROBECAL-TRACE	15.06	21.59	25.35	27.61	29.26	32.86	34.37	35.23
PROBECAL-PROMPT&TRACE	19.92	25.80	28.44	29.92	31.05	33.60	34.73	35.48
Dataset	**Intermediate**							
# Samples	1	2	3	4	5	10	15	20
TROVE	5.40	8.26	10.17	11.49	12.35	14.57	15.64	16.11
TROVE-SAMPLE	5.83	8.82	10.58	11.67	12.76	14.69	15.68	16.18
TROVE-SORT	5.40	8.32	10.20	11.50	12.26	14.44	15.40	15.91
TROVE-E.S.L.	5.40	8.32	10.26	11.51	12.27	14.37	15.28	15.74
PROBECAL-PROMPT	7.20	9.66	11.07	11.97	12.57	14.43	15.30	15.63
PROBECAL-TRACE	5.40	8.28	10.21	11.46	12.33	14.57	15.54	16.07
PROBECAL-PROMPT&TRACE	7.20	9.74	11.23	12.09	12.72	14.59	15.49	15.97
Dataset	**Precalculus**							
# Samples	1	2	3	4	5	10	15	20
TROVE	10.11	13.72	15.55	16.95	17.52	20.07	20.74	21.51
TROVE-SAMPLE	10.43	14.16	15.97	17.03	17.76	20.05	20.57	21.16
TROVE-SORT	10.11	13.76	15.76	16.89	17.55	19.53	20.00	20.75
TROVE-E.S.L.	10.11	13.80	15.64	16.81	17.42	19.16	19.56	20.08
PROBECAL-PROMPT	11.99	15.33	16.89	17.63	18.45	20.16	21.10	21.27
PROBECAL-TRACE	10.11	13.83	15.77	16.94	17.45	19.67	20.25	20.85
PROBECAL-PROMPT&TRACE	11.99	15.43	17.08	17.47	18.23	19.82	20.74	21.11
Dataset	**Number**							
# Samples	1	2	3	4	5	10	15	20
TROVE	13.96	19.44	22.22	23.81	24.84	27.12	27.70	28.30
TROVE-SAMPLE	16.31	21.04	23.53	24.69	25.60	27.42	28.06	28.30
TROVE-SORT	13.96	19.57	22.38	23.90	24.74	26.73	27.18	27.81
TROVE-E.S.L.	13.96	19.62	22.37	23.87	24.52	26.16	26.60	26.97
PROBECAL-PROMPT	16.94	21.17	23.63	24.96	25.59	27.67	28.42	28.93
PROBECAL-TRACE	13.96	19.67	22.29	25.15	26.20	28.75	29.83	30.83
PROBECAL-PROMPT&TRACE	16.94	22.01	24.78	26.26	27.01	29.30	30.43	31.00

Table 10: Dynamic Tool-Using results of different variants on each dataset I

Dataset	**Prealgebra**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	19.66	23.79	26.63	28.31	29.34	31.79	32.81	33.36
ProbeCal-Trace	19.90	25.79	28.25	29.68	30.59	33.01	33.98	34.45
ProbeCal-Prompt&Trace	19.66	25.04	27.80	29.27	30.23	32.52	33.58	34.25
ProbeCal-Prompt-TS	19.58	23.99	26.46	28.26	29.45	31.78	32.85	33.45
ProbeCal-Trace-TS	19.90	25.79	28.25	29.68	30.59	33.00	33.97	34.43
ProbeCal-Prompt&Trace-TS	19.58	25.23	27.70	29.31	30.31	32.44	33.76	34.22
ProbeCal-Prompt-Weight	19.43	23.87	26.53	28.12	29.15	31.73	32.77	33.29
ProbeCal-Trace-Weight	19.90	25.78	28.29	29.68	30.61	33.15	34.12	34.69
ProbeCal-Prompt&Trace-Weight	19.43	25.11	27.70	29.16	30.17	32.58	33.69	34.26
ProbeCal-Prompt-Sort	19.85	23.98	26.64	28.22	29.24	31.73	32.62	33.25
ProbeCal-Trace-Sort	19.90	25.78	28.13	29.65	30.57	33.01	33.94	34.47
ProbeCal-Prompt&Trace-Sort	19.85	25.16	27.72	29.19	30.03	32.59	33.53	34.28
ProbeCal-Prompt-E.S.L.	19.15	23.17	25.73	27.28	28.41	30.72	31.60	32.22
ProbeCal-Trace-E.S.L.	19.90	25.54	27.72	29.19	30.06	32.31	33.45	33.88
ProbeCal-Prompt&Trace-E.S.L.	19.15	24.15	26.45	27.81	28.74	30.97	32.09	32.67
Dataset	**Count**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	20.14	22.79	24.17	24.98	25.61	26.59	27.11	27.28
ProbeCal-Trace	16.97	21.29	23.45	24.34	25.16	26.59	27.32	27.55
ProbeCal-Prompt&Trace	20.14	23.56	24.78	25.52	26.13	27.06	27.63	27.76
ProbeCal-Prompt-TS	20.08	22.43	24.11	25.03	25.56	26.63	27.19	27.30
ProbeCal-Trace-TS	16.97	21.29	23.46	24.35	25.16	26.56	27.30	27.54
ProbeCal-Prompt&Trace-TS	20.08	23.29	24.78	25.41	25.89	26.88	27.38	27.89
ProbeCal-Prompt-Weight	19.99	22.63	24.19	24.94	25.62	26.91	27.30	27.50
ProbeCal-Trace-Weight	16.97	21.35	23.43	24.33	25.07	26.62	27.23	27.58
ProbeCal-Prompt&Trace-Weight	19.99	23.54	24.75	25.56	25.94	27.07	27.70	28.10
ProbeCal-Prompt-Sort	19.89	22.75	24.17	25.14	25.70	26.90	27.39	27.56
ProbeCal-Trace-Sort	16.97	21.28	23.36	24.26	24.96	26.45	27.22	27.42
ProbeCal-Prompt&Trace-Sort	19.89	23.48	24.74	25.44	25.96	27.26	27.70	28.04
ProbeCal-Prompt-E.S.L.	19.63	22.08	23.46	24.31	24.73	25.88	26.38	26.53
ProbeCal-Trace-E.S.L.	16.97	20.94	22.94	23.77	24.59	25.65	26.33	26.67
ProbeCal-Prompt&Trace-E.S.L.	19.63	22.71	23.73	24.57	24.89	25.70	26.16	26.28
Dataset	**TabMWP**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	28.50	34.60	38.08	40.49	42.15	46.16	48.13	48.87
ProbeCal-Trace	21.27	30.23	35.58	39.06	41.54	48.47	51.45	53.40
ProbeCal-Prompt&Trace	28.50	36.82	40.93	43.83	45.87	50.93	53.32	54.34
ProbeCal-Prompt-TS	27.93	34.22	38.31	40.37	42.05	46.23	48.11	48.97
ProbeCal-Trace-TS	21.27	30.23	35.59	39.06	41.55	48.51	51.51	53.50
ProbeCal-Prompt&Trace-TS	27.93	36.39	41.19	43.97	45.89	51.06	53.31	54.50
ProbeCal-Prompt-Weight	28.37	34.32	38.13	40.32	41.98	46.42	48.25	49.02
ProbeCal-Trace-Weight	21.27	30.26	35.70	39.10	41.65	48.64	51.71	53.72
ProbeCal-Prompt&Trace-Weight	28.37	36.60	41.12	43.98	45.93	51.31	53.63	54.74
ProbeCal-Prompt-Sort	28.08	34.33	38.37	40.35	41.74	46.50	48.18	49.10
ProbeCal-Trace-Sort	21.27	30.26	35.63	39.08	41.65	48.69	51.69	53.68
ProbeCal-Prompt&Trace-Sort	28.08	36.67	41.24	44.01	45.81	51.33	53.66	54.97
ProbeCal-Prompt-E.S.L.	27.30	33.23	37.10	39.18	41.02	45.36	47.34	48.40
ProbeCal-Trace-E.S.L.	21.27	30.15	35.32	38.60	41.05	47.39	50.25	51.98
ProbeCal-Prompt&Trace-E.S.L.	27.30	35.30	39.73	42.17	44.35	49.29	51.48	52.61
Dataset	**Geometry**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	4.67	5.99	6.70	7.32	7.66	8.88	9.58	10.01
ProbeCal-Trace	3.99	5.44	6.33	6.89	7.40	8.41	8.80	9.26
ProbeCal-Prompt&Trace	4.67	6.23	6.86	7.56	7.76	8.70	9.28	9.61
ProbeCal-Prompt-TS	4.65	5.82	6.67	7.11	7.75	8.94	9.61	10.12
ProbeCal-Trace-TS	3.99	5.44	6.33	6.88	7.37	8.38	8.79	9.26
ProbeCal-Prompt&Trace-TS	4.65	6.15	6.83	7.32	7.99	8.81	9.23	9.58
ProbeCal-Prompt-Weight	4.55	5.82	6.75	7.20	7.51	8.81	9.31	9.77
ProbeCal-Trace-Weight	3.99	5.46	6.32	6.94	7.45	8.46	8.72	9.27
ProbeCal-Prompt&Trace-Weight	4.55	6.10	7.05	7.41	7.65	8.67	9.04	9.47
ProbeCal-Prompt-Sort	4.55	6.00	6.70	7.09	7.67	9.01	9.82	10.51
ProbeCal-Trace-Sort	3.99	5.44	6.16	6.81	7.28	8.19	8.48	9.03
ProbeCal-Prompt&Trace-Sort	4.55	6.18	6.86	7.32	7.62	8.50	8.91	9.49
ProbeCal-Prompt-E.S.L.	3.80	4.68	5.37	5.53	6.11	7.08	7.78	8.15
ProbeCal-Trace-E.S.L.	3.99	5.25	5.81	6.36	6.70	6.81	6.76	6.72
ProbeCal-Prompt&Trace-E.S.L.	3.80	4.65	5.20	5.34	5.61	5.92	6.38	6.53

Table 11: Dynamic Tool-Using results of different variants on each dataset II

Dataset	**Algebra**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	19.92	25.13	27.66	29.13	30.17	32.78	33.83	34.42
ProbeCal-Trace	15.06	21.59	25.35	27.61	29.26	32.86	34.37	35.23
ProbeCal-Prompt&Trace	19.92	25.80	28.44	29.92	31.05	33.60	34.73	35.48
ProbeCal-Prompt-TS	19.84	25.07	27.58	29.20	30.28	32.82	33.90	34.55
ProbeCal-Trace-TS	15.06	21.59	25.34	27.61	29.26	32.84	34.35	35.21
ProbeCal-Prompt&Trace-TS	19.84	25.69	28.41	29.97	31.11	33.69	34.82	35.55
ProbeCal-Prompt-Weight	19.89	25.09	27.53	29.05	30.33	32.65	33.78	34.41
ProbeCal-Trace-Weight	15.06	21.56	25.33	27.62	29.26	32.89	34.31	35.24
ProbeCal-Prompt&Trace-Weight	19.89	25.75	28.34	29.87	31.06	33.59	34.69	35.50
ProbeCal-Prompt-Sort	20.15	25.20	27.41	29.12	30.08	32.56	33.66	34.17
ProbeCal-Trace-Sort	15.06	21.55	25.30	27.59	29.20	32.69	34.17	34.97
ProbeCal-Prompt&Trace-Sort	20.15	25.87	28.26	29.85	30.94	33.41	34.56	35.20
ProbeCal-Prompt-E.S.L.	18.78	23.67	26.11	27.92	29.03	31.59	32.84	33.53
ProbeCal-Trace-E.S.L.	15.06	21.40	25.01	27.26	28.82	32.20	33.66	34.44
ProbeCal-Prompt&Trace-E.S.L.	18.78	24.14	26.73	28.51	29.62	32.29	33.54	34.26
Dataset	**Intermediate**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	7.20	9.66	11.07	11.97	12.57	14.43	15.30	15.63
ProbeCal-Trace	5.40	8.28	10.21	11.46	12.33	14.57	15.54	16.07
ProbeCal-Prompt&Trace	7.20	9.74	11.23	12.09	12.72	14.59	15.49	15.97
ProbeCal-Prompt-TS	7.21	9.69	11.05	11.99	12.72	14.57	15.34	15.76
ProbeCal-Trace-TS	5.40	8.28	10.21	11.46	12.33	14.58	15.53	16.07
ProbeCal-Prompt&Trace-TS	7.21	9.74	11.18	12.14	12.83	14.61	15.47	15.91
ProbeCal-Prompt-Weight	7.27	9.73	10.99	11.86	12.56	14.22	15.01	15.39
ProbeCal-Trace-Weight	5.40	8.30	10.21	11.47	12.33	14.63	15.56	16.20
ProbeCal-Prompt&Trace-Weight	7.27	9.84	11.15	12.11	12.71	14.41	15.26	15.73
ProbeCal-Prompt-Sort	7.05	9.39	10.69	11.59	12.21	14.00	14.99	15.27
ProbeCal-Trace-Sort	5.40	8.27	10.24	11.50	12.38	14.61	15.58	16.02
ProbeCal-Prompt&Trace-Sort	7.05	9.51	10.92	11.79	12.37	14.31	15.25	15.74
ProbeCal-Prompt-E.S.L.	6.34	8.33	9.60	10.57	11.26	13.22	14.08	14.53
ProbeCal-Trace-E.S.L.	5.40	8.29	10.20	11.44	12.26	14.47	15.29	15.66
ProbeCal-Prompt&Trace-E.S.L.	6.34	8.42	9.62	10.62	11.29	13.24	14.21	14.63
Dataset	**Precalculus**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	11.99	15.33	16.89	17.63	18.45	20.16	21.10	21.27
ProbeCal-Trace	10.11	13.83	15.77	16.94	17.45	19.67	20.25	20.85
ProbeCal-Prompt&Trace	11.99	15.43	17.08	17.47	18.23	19.82	20.74	21.11
ProbeCal-Prompt-TS	11.88	15.41	17.09	17.81	18.41	20.12	20.94	21.20
ProbeCal-Trace-TS	10.11	13.83	15.77	16.95	17.43	19.70	20.26	20.87
ProbeCal-Prompt&Trace-TS	11.88	15.61	17.11	17.61	18.30	19.83	20.61	21.02
ProbeCal-Prompt-Weight	11.96	15.01	16.64	17.53	18.43	19.94	20.69	21.11
ProbeCal-Trace-Weight	10.11	13.77	15.72	16.97	17.49	19.66	20.25	21.10
ProbeCal-Prompt&Trace-Weight	11.96	15.27	16.88	17.68	18.41	20.01	20.66	21.33
ProbeCal-Prompt-Sort	11.72	15.05	16.62	17.64	18.47	20.43	21.34	21.82
ProbeCal-Trace-Sort	10.11	13.83	15.67	16.99	17.38	19.81	20.27	20.98
ProbeCal-Prompt&Trace-Sort	11.72	15.27	16.63	17.49	18.20	19.90	20.84	21.34
ProbeCal-Prompt-E.S.L.	11.30	14.76	16.07	16.99	17.23	18.50	19.19	19.45
ProbeCal-Trace-E.S.L.	10.11	13.57	15.41	16.43	16.93	18.97	19.35	20.14
ProbeCal-Prompt&Trace-E.S.L.	11.30	14.63	15.70	16.49	16.86	17.94	18.23	18.75
Dataset	**Number**							
# Samples	1	2	3	4	5	10	15	20
ProbeCal-Prompt	16.94	21.17	23.63	24.96	25.59	27.67	28.42	28.93
ProbeCal-Trace	13.96	20.07	23.29	25.15	26.20	28.75	29.83	30.83
ProbeCal-Prompt&Trace	16.94	22.01	24.78	26.26	27.01	29.30	30.43	31.00
ProbeCal-Prompt-TS	16.81	21.45	23.53	24.87	25.70	27.64	28.44	28.89
ProbeCal-Trace-TS	13.96	20.07	23.29	25.15	26.20	28.76	29.86	30.83
ProbeCal-Prompt&Trace-TS	16.81	22.28	24.75	26.24	27.07	29.44	30.51	31.09
ProbeCal-Prompt-Weight	17.37	22.00	24.11	25.17	26.04	28.11	28.72	29.21
ProbeCal-Trace-Weight	13.96	20.10	23.33	25.24	26.28	28.87	29.89	30.87
ProbeCal-Prompt&Trace-Weight	17.37	22.93	25.29	26.49	27.50	29.82	30.73	31.27
ProbeCal-Prompt-Sort	16.78	21.21	23.58	24.84	25.72	27.60	28.46	28.86
ProbeCal-Trace-Sort	13.96	20.07	23.19	25.12	26.16	28.69	29.67	30.70
ProbeCal-Prompt&Trace-Sort	16.78	22.05	24.65	25.98	26.81	29.09	30.23	30.78
ProbeCal-Prompt-E.S.L.	15.61	19.80	21.85	23.13	23.94	25.96	26.88	27.20
ProbeCal-Trace-E.S.L.	13.96	19.92	22.95	24.65	25.64	27.74	28.67	29.34
ProbeCal-Prompt&Trace-E.S.L.	15.61	20.28	22.52	23.80	24.58	26.51	27.59	28.21

Table 12: ECE loss on the static tool using settings on each dataset. The settings are denoted using two letters, the first letter corresponds to whether to use weighted training, with 'W' representing yes and 'N' representing no. The second letter corresponds to whether and in which way using LLM logit as the input or not, with 'S' representing using SORT to get the logit, 'E' representing using E.S.L. to get the logit, and 'N' representing not using LLM logit. When there are two numbers for the result of a setting, the first number denotes the training ECE loss and the second number denotes the testing ECE loss. When there is one number for the result of a setting, the number refers to the testing ECE loss.

Settings	NN	WN	NS	WS	NE	WE
Dataset			**Prealgebra**			
LLM Logit	0.730, 0.749	0.730, 0.749	0.345, 0.326	0.345, 0.326	0.149, 0.055	0.148, 0.055
PROBECAL-PROMPT	0.012, 0.094	0.013, 0.092	0.014, 0.093	0.015, 0.094	0.096, 0.154	0.099, 0.153
PROBECAL-PROMPT(TS)	0.022, 0.081	0.022, 0.082	0.026, 0.081	0.026, 0.082	0.114, 0.148	0.111, 0.150
PROBECAL-TRACE	0.013, 0.088	0.021, 0.081	0.018, 0.083	0.021, 0.078	0.108, 0.136	0.115, 0.139
PROBECAL-TRACE(TS)	0.010, 0.094	0.038, 0.111	0.011, 0.092	0.013, 0.089	0.100, 0.138	0.104, 0.142
PROBECAL-PROMPT&TRACE	0.055	0.046	0.052	0.049	0.130	0.131
Dataset			**Count**			
LLM Logit	0.841, 0.804	0.841, 0.804	0.515, 0.470	0.516, 0.470	0.156, 0.095	0.156, 0.095
PROBECAL-PROMPT	0.010, 0.076	0.013, 0.070	0.010, 0.078	0.012, 0.075	0.074, 0.125	0.078, 0.128
PROBECAL-PROMPT(TS)	0.014, 0.069	0.014, 0.068	0.012, 0.075	0.014, 0.072	0.082, 0.126	0.085, 0.129
PROBECAL-TRACE	0.009, 0.072	0.011, 0.066	0.010, 0.072	0.014, 0.066	0.082, 0.118	0.085, 0.117
PROBECAL-TRACE(TS)	0.007, 0.079	0.007, 0.074	0.008, 0.077	0.008, 0.078	0.076, 0.118	0.108, 0.134
PROBECAL-PROMPT&TRACE	0.051	0.047	0.055	0.049	0.121	0.119
Dataset			**TabMWP**			
LLM Logit	0.632, 0.690	0.632, 0.690	0.301, 0.305	0.301, 0.305	0.303, 0.163	0.304, 0.163
PROBECAL-PROMPT	0.009, 0.067	0.012, 0.062	0.013, 0.064	0.012, 0.067	0.131, 0.166	0.130, 0.165
PROBECAL-PROMPT(TS)	0.019, 0.054	0.018, 0.053	0.021, 0.054	0.023, 0.054	0.145, 0.170	0.139, 0.168
PROBECAL-TRACE	0.010, 0.057	0.014, 0.052	0.012, 0.050	0.015, 0.049	0.135, 0.144	0.140, 0.145
PROBECAL-TRACE(TS)	0.009, 0.060	0.009, 0.060	0.010, 0.053	0.011, 0.055	0.151, 0.152	0.154, 0.154
PROBECAL-PROMPT&TRACE	0.035	0.033	0.038	0.040	0.163	0.160
Dataset			**Geometry**			
LLM Logit	0.957, 0.925	0.957, 0.925	0.190, 0.284	0.189, 0.284	0.044, 0.044	0.044, 0.044
PROBECAL-PROMPT	0.006, 0.041	0.010, 0.036	0.006, 0.046	0.008, 0.042	0.229, 0.198	0.235, 0.206
PROBECAL-PROMPT(TS)	0.009, 0.036	0.011, 0.034	0.009, 0.042	0.011, 0.039	0.239, 0.209	0.242, 0.213
PROBECAL-TRACE	0.005, 0.036	0.010, 0.029	0.007, 0.040	0.006, 0.039	0.229, 0.203	0.231, 0.204
PROBECAL-TRACE(TS)	0.006, 0.034	0.006, 0.033	0.005, 0.041	0.005, 0.040	0.240, 0.215	0.247, 0.223
PROBECAL-PROMPT&TRACE	0.034	0.028	0.037	0.037	0.194	0.200
Dataset			**Algebra**			
LLM Logit	0.814, 0.823	0.814, 0.823	0.428, 0.407	0.428, 0.407	0.119, 0.048	0.119, 0.048
PROBECAL-PROMPT	0.009, 0.052	0.010, 0.050	0.010, 0.053	0.014, 0.049	0.095, 0.126	0.101, 0.129
PROBECAL-PROMPT(TS)	0.019, 0.041	0.018, 0.041	0.019, 0.044	0.022, 0.042	0.110, 0.131	0.107, 0.131
PROBECAL-TRACE	0.010, 0.047	0.011, 0.046	0.011, 0.048	0.015, 0.040	0.102, 0.118	0.108, 0.123
PROBECAL-TRACE(TS)	0.007, 0.051	0.006, 0.052	0.007, 0.053	0.007, 0.049	0.104, 0.119	0.109, 0.125
PROBECAL-PROMPT&TRACE	0.026	0.024	0.029	0.025	0.121	0.125
Dataset			**Intermediate**			
LLM Logit	0.916, 0.936	0.916, 0.936	0.448, 0.435	0.449, 0.435	0.065, 0.043	0.065, 0.043
PROBECAL-PROMPT	0.007, 0.032	0.010, 0.026	0.007, 0.033	0.012, 0.033	0.094, 0.118	0.107, 0.137
PROBECAL-PROMPT(TS)	0.011, 0.030	0.012, 0.026	0.012, 0.031	0.015, 0.032	0.104, 0.126	0.114, 0.142
PROBECAL-TRACE	0.008, 0.038	0.007, 0.035	0.009, 0.037	0.008, 0.035	0.102, 0.132	0.117, 0.144
PROBECAL-TRACE(TS)	0.004, 0.042	0.006, 0.038	0.005, 0.044	0.005, 0.040	0.115, 0.142	0.122, 0.147
PROBECAL-PROMPT&TRACE	0.025	0.022	0.027	0.028	0.127	0.141
Dataset			**Precalculus**			
LLM Logit	0.898, 0.884	0.897, 0.884	0.346, 0.323	0.347, 0.323	0.085, 0.042	0.084, 0.042
PROBECAL-PROMPT	0.011, 0.058	0.011, 0.059	0.010, 0.063	0.013, 0.060	0.145, 0.177	0.152, 0.182
PROBECAL-PROMPT(TS)	0.020, 0.051	0.014, 0.057	0.016, 0.057	0.016, 0.059	0.157, 0.183	0.165, 0.191
PROBECAL-TRACE	0.014, 0.056	0.013, 0.053	0.013, 0.054	0.016, 0.054	0.150, 0.182	0.167, 0.193
PROBECAL-TRACE(TS)	0.012, 0.058	0.012, 0.062	0.012, 0.061	0.013, 0.062	0.160, 0.189	0.183, 0.207
PROBECAL-PROMPT&TRACE	0.041	0.043	0.043	0.042	0.172	0.186
Dataset			**Number**			
LLM Logit	0.758, 0.762	0.758, 0.762	0.368, 0.366	0.368, 0.366	0.137, 0.067	0.137, 0.067
PROBECAL-PROMPT	0.011, 0.073	0.015, 0.068	0.012, 0.074	0.013, 0.074	0.087, 0.123	0.096, 0.125
PROBECAL-PROMPT(TS)	0.016, 0.066	0.022, 0.060	0.020, 0.064	0.023, 0.063	0.100, 0.122	0.112, 0.125
PROBECAL-TRACE	0.011, 0.068	0.016, 0.063	0.017, 0.059	0.019, 0.055	0.101, 0.119	0.103, 0.121
PROBECAL-TRACE(TS)	0.008, 0.071	0.011, 0.070	0.010, 0.067	0.011, 0.063	0.098, 0.122	0.100, 0.122
PROBECAL-PROMPT&TRACE	0.044	0.038	0.038	0.037	0.107	0.114

Table 13: ECE loss on the dynamic tool using settings on each dataset. The settings are denoted using two letters, the first letter corresponds to whether to use weighted training, with 'W' representing yes and 'N' representing no. The second letter corresponds to whether and in which way using LLM logit as the input or not, with 'S' representing using SORT to get the logit, 'E' representing using E.S.L. to get the logit, and 'N' representing not using LLM logit. When there are two numbers for the result of a setting, the first number denotes the training ECE loss and the second number denotes the testing ECE loss. When there is one number for the result of a setting, the number refers to the testing ECE loss.

Settings	NN	WN	NS	WS	NE	WE
Dataset			**Prealgebra**			
LLM Logit	0.766, 0.801	0.767, 0.801	0.293, 0.326	0.293, 0.326	0.088, 0.064	0.088, 0.064
PROBECAL-PROMPT	0.009, 0.079	0.012, 0.077	0.008, 0.079	0.015, 0.071	0.076, 0.134	0.084, 0.139
PROBECAL-PROMPT(TS)	0.014, 0.073	0.014, 0.075	0.013, 0.072	0.014, 0.072	0.086, 0.132	0.087, 0.138
PROBECAL-TRACE	0.009, 0.092	0.012, 0.084	0.011, 0.091	0.013, 0.088	0.080, 0.121	0.086, 0.122
PROBECAL-TRACE(TS)	0.006, 0.094	0.006, 0.092	0.006, 0.098	0.006, 0.096	0.074, 0.127	0.080, 0.125
PROBECAL-PROMPT&TRACE	0.049	0.045	0.048	0.044	0.115	0.121
Dataset			**Count**			
LLM Logit	0.856, 0.831	0.855, 0.831	0.437, 0.435	0.437, 0.435	0.129, 0.138	0.129, 0.138
PROBECAL-PROMPT	0.006, 0.051	0.009, 0.048	0.007, 0.050	0.009, 0.048	0.078, 0.104	0.090, 0.109
PROBECAL-PROMPT(TS)	0.009, 0.045	0.009, 0.048	0.008, 0.048	0.009, 0.050	0.089, 0.104	0.087, 0.110
PROBECAL-TRACE	0.007, 0.052	0.010, 0.049	0.006, 0.054	0.009, 0.052	0.070, 0.093	0.075, 0.096
PROBECAL-TRACE(TS)	0.005, 0.055	0.005, 0.057	0.004, 0.058	0.005, 0.058	0.071, 0.093	0.073, 0.098
PROBECAL-PROMPT&TRACE	0.034	0.031	0.036	0.032	0.095	0.098
Dataset			**TabMWP**			
LLM Logit	0.768, 0.788	0.768, 0.788	0.341, 0.387	0.342, 0.387	0.187, 0.148	0.186, 0.148
PROBECAL-PROMPT	0.007, 0.078	0.013, 0.074	0.009, 0.076	0.010, 0.079	0.046, 0.098	0.048, 0.102
PROBECAL-PROMPT(TS)	0.013, 0.070	0.014, 0.072	0.012, 0.070	0.012, 0.076	0.052, 0.096	0.050, 0.101
PROBECAL-TRACE	0.011, 0.072	0.011, 0.071	0.011, 0.075	0.015, 0.071	0.059, 0.083	0.055, 0.081
PROBECAL-TRACE(TS)	0.006, 0.079	0.007, 0.077	0.006, 0.082	0.005, 0.085	0.058, 0.084	0.046, 0.084
PROBECAL-PROMPT&TRACE	0.043	0.039	0.043	0.042	0.083	0.081
Dataset			**Geometry**			
LLM Logit	0.977, 0.961	0.977, 0.961	0.260, 0.314	0.260, 0.314	0.068, 0.052	0.068, 0.052
PROBECAL-PROMPT	0.003, 0.025	0.006, 0.021	0.003, 0.023	0.006, 0.024	0.137, 0.117	0.161, 0.141
PROBECAL-PROMPT(TS)	0.003, 0.024	0.005, 0.022	0.004, 0.023	0.005, 0.024	0.159, 0.137	0.176, 0.152
PROBECAL-TRACE	0.004, 0.024	0.006, 0.019	0.005, 0.022	0.006, 0.019	0.151, 0.127	0.157, 0.132
PROBECAL-TRACE(TS)	0.003, 0.025	0.004, 0.022	0.004, 0.025	0.003, 0.023	0.173, 0.147	0.182, 0.155
PROBECAL-PROMPT&TRACE	0.021	0.016	0.020	0.017	0.123	0.136
Dataset			**Algebra**			
LLM Logit	0.846, 0.851	0.846, 0.851	0.401, 0.396	0.400, 0.396	0.103, 0.096	0.103, 0.096
PROBECAL-PROMPT	0.006, 0.054	0.009, 0.049	0.005, 0.055	0.011, 0.048	0.070, 0.107	0.076, 0.109
PROBECAL-PROMPT(TS)	0.010, 0.049	0.009, 0.048	0.009, 0.049	0.010, 0.050	0.085, 0.108	0.079, 0.109
PROBECAL-TRACE	0.007, 0.051	0.010, 0.049	0.009, 0.050	0.008, 0.051	0.073, 0.100	0.079, 0.103
PROBECAL-TRACE(TS)	0.004, 0.055	0.004, 0.057	0.005, 0.055	0.004, 0.055	0.070, 0.101	0.073, 0.104
PROBECAL-PROMPT&TRACE	0.027	0.025	0.028	0.025	0.100	0.105
Dataset			**Intermediate**			
LLM Logit	0.934, 0.945	0.934, 0.945	0.345, 0.357	0.346, 0.357	0.025, 0.036	0.025, 0.036
PROBECAL-PROMPT	0.005, 0.027	0.007, 0.026	0.005, 0.028	0.007, 0.028	0.085, 0.101	0.096, 0.111
PROBECAL-PROMPT(TS)	0.006, 0.025	0.007, 0.025	0.007, 0.025	0.006, 0.028	0.107, 0.116	0.104, 0.116
PROBECAL-TRACE	0.003, 0.029	0.007, 0.026	0.004, 0.027	0.008, 0.023	0.081, 0.093	0.091, 0.101
PROBECAL-TRACE(TS)	0.003, 0.029	0.003, 0.031	0.003, 0.027	0.004, 0.029	0.085, 0.094	0.096, 0.105
PROBECAL-PROMPT&TRACE	0.020	0.019	0.021	0.019	0.097	0.107
Dataset			**Precalculus**			
LLM Logit	0.913, 0.902	0.913, 0.902	0.272, 0.305	0.273, 0.305	0.041, 0.041	0.041, 0.041
PROBECAL-PROMPT	0.007, 0.040	0.009, 0.039	0.007, 0.041	0.011, 0.038	0.116, 0.118	0.138, 0.136
PROBECAL-PROMPT(TS)	0.010, 0.036	0.008, 0.039	0.011, 0.038	0.011, 0.038	0.126, 0.124	0.143, 0.139
PROBECAL-TRACE	0.009, 0.046	0.012, 0.042	0.008, 0.047	0.011, 0.043	0.150, 0.145	0.149, 0.135
PROBECAL-TRACE(TS)	0.007, 0.050	0.008, 0.052	0.007, 0.049	0.008, 0.048	0.163, 0.150	0.168, 0.143
PROBECAL-PROMPT&TRACE	0.027	0.027	0.030	0.031	0.128	0.142
Dataset			**Number**			
LLM Logit	0.808, 0.861	0.808, 0.861	0.290, 0.306	0.290, 0.306	0.066, 0.036	0.065, 0.036
PROBECAL-PROMPT	0.007, 0.032	0.009, 0.026	0.008, 0.035	0.013, 0.026	0.081, 0.126	0.087, 0.128
PROBECAL-PROMPT(TS)	0.010, 0.028	0.011, 0.026	0.011, 0.031	0.012, 0.027	0.089, 0.128	0.085, 0.127
PROBECAL-TRACE	0.007, 0.040	0.013, 0.031	0.008, 0.041	0.010, 0.036	0.084, 0.118	0.085, 0.117
PROBECAL-TRACE(TS)	0.005, 0.042	0.005, 0.041	0.005, 0.044	0.006, 0.040	0.087, 0.119	0.083, 0.116
PROBECAL-PROMPT&TRACE	0.021	0.020	0.024	0.024	0.130	0.133

Table 14: Experiment results for TabMWP in static tool-using settings using `CODELLAMA-13B-INSTRUCT-hf` with different size of training set from 50 to 500.

Method	Size 50		Size 100		Size 200		Size 500	
	Acc	ECE	Acc	ECE	Acc	ECE	Acc	ECE
Static Tool-Using								
INSTANCE	65.66	0.571	65.66	0.571	65.66	0.571	65.66	0.571
PROBECAL-PROMPT	66.19	0.117	67.60	0.088	68.06	0.084	68.11	0.077
PROBECAL-TRACE	67.72	0.076	68.50	0.071	69.55	0.055	70.46	0.034
PROBECAL-PROMPT&TRACE	67.07	0.064	69.10	0.042	70.28	0.038	71.14	0.034

Table 15: Experiment results for TabMWP in static tool-using setting using `CODELLAMA-13B-INSTRUCT-hf` and `Llama3-8b-Instruct` (Size 500).

Method	`CODELLAMA-13B-Inst`			`Llama3-8b-Inst`		
	Acc@5	Acc@10	ECE	Acc@5	Acc@10	ECE
INSTANCE	62.94	65.66	0.571	73.07	75.20	0.404
PROBECAL-PROMPT	66.59	68.11	0.077	74.79	76.53	0.080
PROBECAL-TRACE	67.09	70.46	0.034	76.05	78.25	0.035
PROBECAL-PROMPT&TRACE	69.01	71.14	0.034	76.79	78.48	0.038

Table 16: Experiment results for TabMWP in static tool-using setting with `CODELLAMA-13B-INSTRUCT-hf` using verbal confidence.

Method	`CODELLAMA-13B-Inst` (Size 50)		
	Acc@5	Acc@10	ECE
INSTANCE	62.94	65.66	0.571
PROBECAL-PROMPT	64.27	66.19	0.117
PROBECAL-TRACE	64.81	67.72	0.076
PROBECAL-PROMPT&TRACE	65.23	67.07	0.064
VERBAL CONFIDENCE	61.52	64.68	0.212

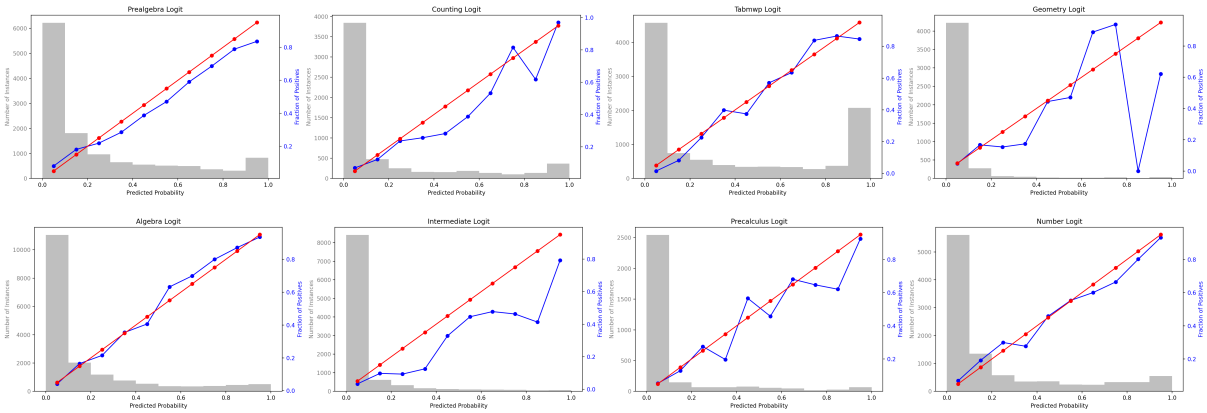

Figure 5: Calibration curve of PROBECAL-PROMPT&TRACE logits of LLM in Static Tool-Using

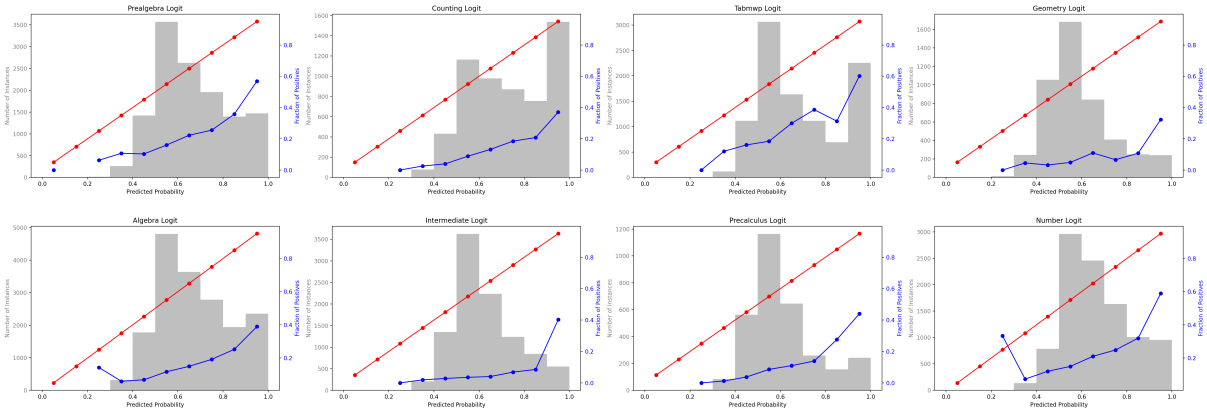

Figure 6: Calibration curve of SORT logits of LLM in Static Tool-Using

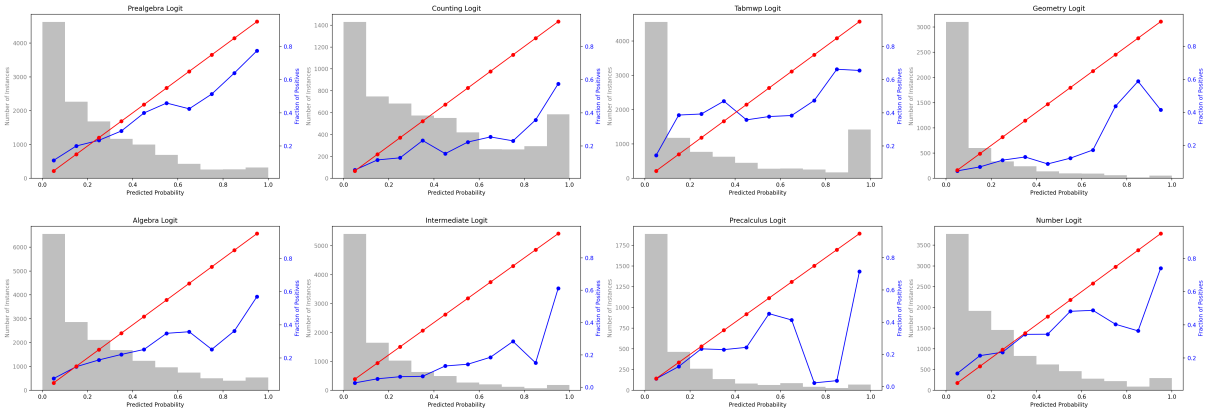

Figure 7: Calibration curve of E.S.L. logits of LLM in Static Tool-Using

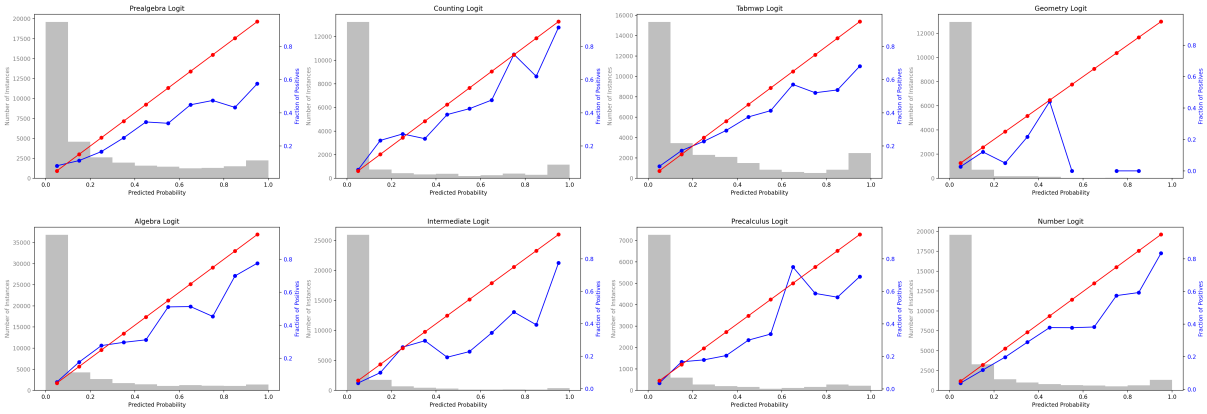

Figure 8: Calibration curve of PROBECAL-PROMPT logits of LLM in Dynamic Tool-Using

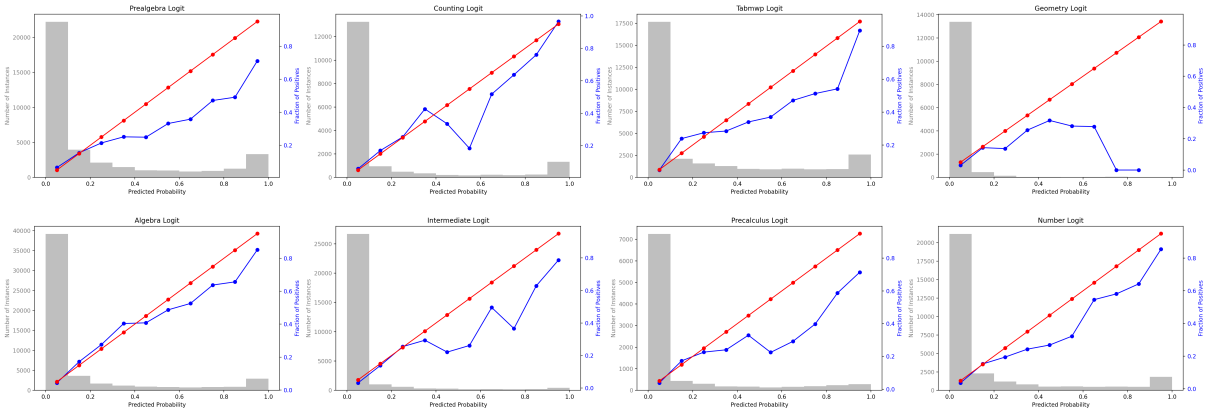

Figure 9: Calibration curve of PROBECAL-TRACE logits of LLM in Dynamic Tool-Using

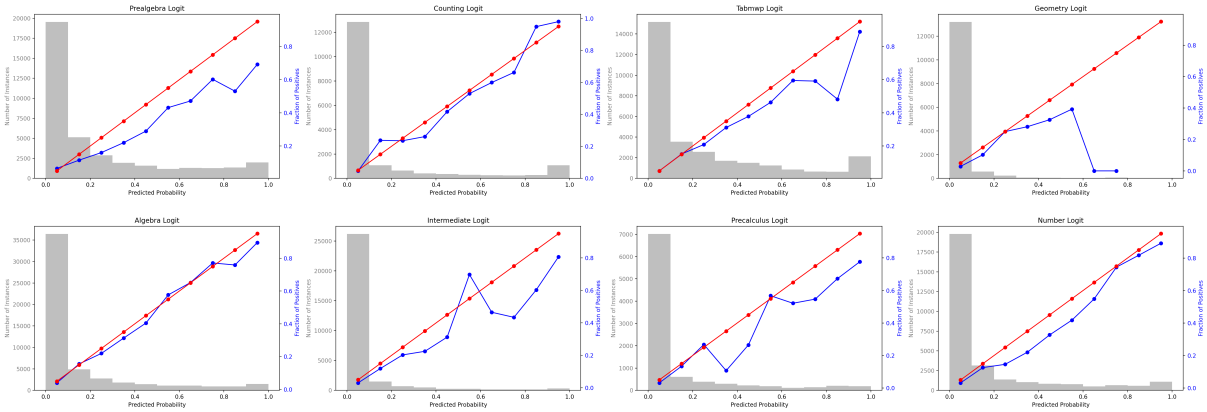

Figure 10: Calibration curve of PROBECAL-PROMPT&TRACE logits of LLM in Dynamic Tool-Using

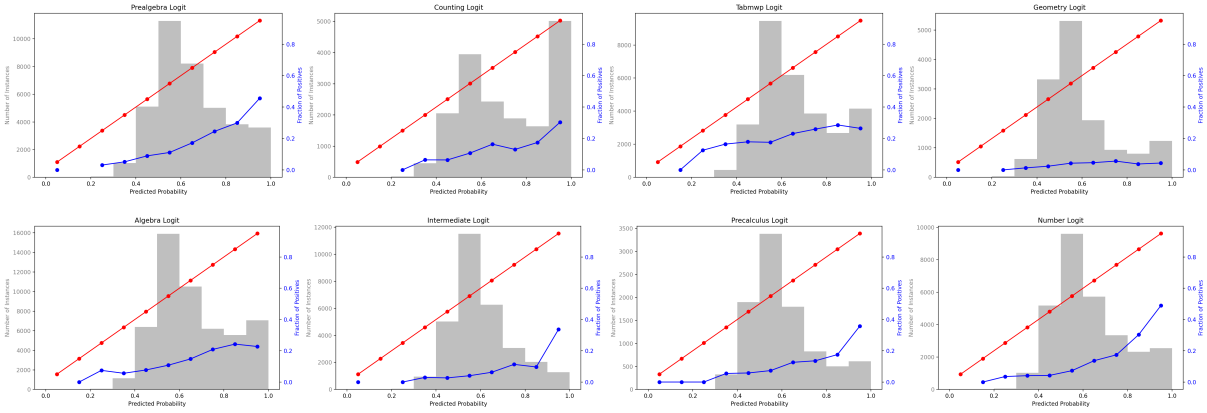

Figure 11: Calibration curve of SORT logits of LLM in Dynamic Tool-Using

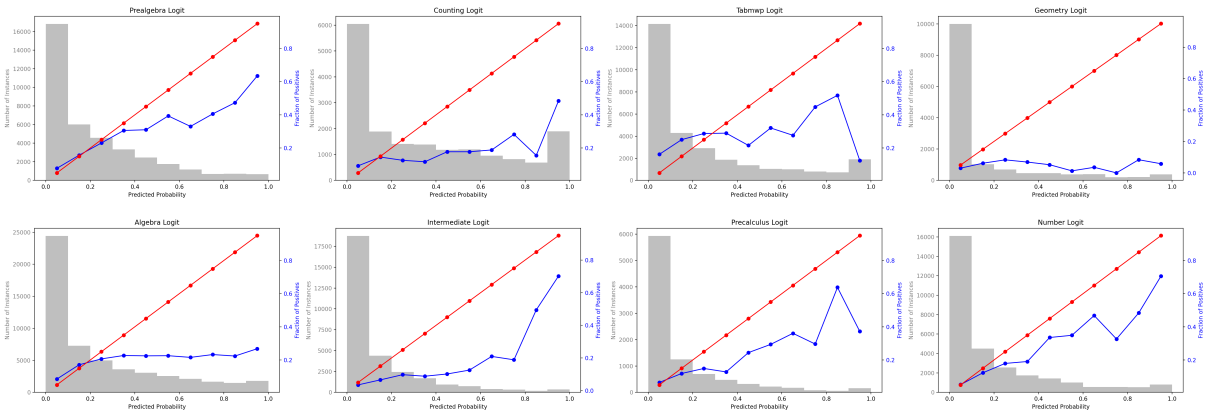

Figure 12: Calibration curve of E.S.L. logits of LLM in Dynamic Tool-Using