Assessing the Belief Consistency of Large Language Models on the Logical Conversation Process

Tomoki Tsujimura, Matīss Rikters, Masaki Asada, Shusaku Egami, Tatsuya Ishigaki, Ken Yano, Hiroya Takamura
National Institute of Advanced Industrial Science and Technology (AIST)
{tsujimura.res, matiss.rikters, masaki.asada, s-egami, ishigaki.tatsuya, yano.ken, takamura.hiroya}@aist.go.jp

Abstract

To reliably interpret the evolving context of an LLM as a reasoning trace, the underlying belief of the LLM needs to transition consistently with the progression of the context. We focus on evaluating whether the beliefs held by a model remain consistent before and after the extension of the context. Previous research on consistency evaluation typically uses datasets with ground-truth answers, which is problematic because task-solving ability acts as a confounding factor, obscuring the direct evaluation of consistency. Furthermore, evaluating cases where inconsistency stems from multiple errors poses difficulties. We propose a new evaluation method to assess the consistency of LLMs in a multiple-choice question answering format, designed so that any option chosen is correct, allowing for the evaluation of the proposed belief consistency. It also supports isolation of errors such as reasoning failures and biases. We reveal that the belief consistency does not improve solely with model size scaling, whereas continual pre-training on code and mathematics text improves it. Furthermore, models trained on code and mathematics text show a seemingly contradictory result of increased logical failures, indicating that belief consistency and superficial consistency are not necessarily directly linked.

1 Introduction

Large language models (LLMs) have been shown remarkable ability to understand given contexts. Building on this capability, many studies have explored techniques that allow LLMs to mimic human thinking processes, such as Chain-of-Thought (CoT) prompting and reasoning models (Wei et al., 2022; Guo et al., 2025; Yang et al., 2025a). While many studies attempt to mimic human thinking processes in LLMs, the mechanisms underlying contextual understanding in LLMs differ from those of humans. Consider the game of *20 Questions* (Bertolazzi et al., 2023; Zhang et al., 2024) as an illustrative example. 20 Questions is a two-player game consisting of a questioner and a guesser. At the beginning of the game, the questioner secretly imagines an entity (e.g., "*apple*"). The guesser may ask yes/no questions (e.g., "*Is it a fruit?*"), and the questioner must respond truthfully and consistently with respect to the initially chosen entity. The goal of the guesser is to identify the entity within 20 questions. When the questioner is human, the entity initially conceived is expected to remain consistent throughout the game. Now consider conducting this game with an LLM acting as the questioner, instructed to imagine an entity without explicitly revealing it in its outputs. In this setting, the LLM can still produce responses to the guesser's questions. However, unlike a human questioner, there is no guarantee that the entity implicitly treated as the correct answer remains consistent across turns. For instance, the LLM may behave as if "*apple*" were the correct answer before a certain question, but switch to behaving as if "*orange*" were the initially chosen entity after the question. Furthermore, since outputs of an LLM are decoded via sampling, the imagined correct answer should not be regarded as a single fixed entity, but rather as the tendency over possible answers.

We focus on such output tendencies and treat them as beliefs internally held by the LLM. If beliefs of an LLM are consistent under successive extensions of the context, the correct answer in the game of 20 Questions would remain unchanged after each question. More generally, beyond the setting, when a model with consistent beliefs is prompted to perform CoT reasoning, we can expect its final belief to transition from the initial belief in accordance with the intermediate reasoning process, allowing the generated reasoning tokens to be interpreted as a trace of actual state transitions. In contrast, for models lacking such belief consistency, even when a shift in reasoning or decision-making

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 40032–40055

is explicitly express in the output, it does not necessarily indicate a genuine shift from the prior output tendencies.

In this study, we propose a method to evaluate the consistency between the beliefs at the prior input context and the final generated outputs. Our evaluation is conducted through a multiple-choice question answering (MCQA) task (Pezeshkpour and Hruschka, 2024). We define the beliefs of a model as its tendencies over the available choices, represented as a probability distribution. When generating answers, we also provide additional context that induces logical reasoning and compare the resulting beliefs with those obtained without such context. We quantitatively evaluate the belief consistency using the Jensen–Shannon (JS) divergence (Lin, 1991). A major concern with this method is that typical MCQA tasks contain a correct answer, which can introduce substantial bias into the model's selection tendencies. To address this issue, we design a task inspired by the game of 20 Questions. In this task setting, all options are equally valid choices, thereby removing the bias of a correct answer. To further mitigate known biases such as likelihood bias (Ohi et al., 2024) and position bias (Pezeshkpour and Hruschka, 2024), the dataset is constructed from options with comparable likelihood levels, and consistency is measured as an average over all permutations of option orderings. Moreover, to examine the behavior of a language model with as little bias as possible, we sample answers via free-form generation rather than computing log probabilities over a pre-defined answer set.

Through experiments across various model families, we found that consistency does not improve through model scaling alone. In contrast, training on mathematics and coding corpora substantially improves consistency and modest gains from scaling. Furthermore, continually pre-trained models on such corpora exhibit higher selection entropy than their corresponding base models.

Our contributions are summarized as follows:

- We propose a quantitative evaluation method and a synthetic dataset to assess the consistency of an LLM's belief. The proposed method employs a 20 Questions-inspired task design in which any option is valid, thereby eliminating correct-answer bias.
- Through experiments with various open-weight models, we find that belief consistency is not improved by model size scaling alone, but is substantially enhanced by continual pre-training on coding and mathematical corpora.

2 Related Work

Currently, research on the consistency of beliefs not explicitly appearing in LLM outputs is still developing, and it remains unclear to what extent it can be expected or how it should be measured. On the other hand, various prior studies exist regarding the consistency of content that does explicitly appear in inputs and outputs (Cheng et al., 2025). Our research differs in its focus on the part of belief that does not directly appear in inputs nor outputs. While much prior research exists on extracting LLM preferences and knowledge, these studies do not primarily focus on examining the consistency of the model's belief (Rafailov et al., 2023; Metropolitansky and Larson, 2025; Gekhman et al., 2025). However, if belief consistency does not hold, it becomes difficult to evaluate the validity of the extracted preferences and knowledge. A theme close to our research is consistency training (Irpan et al., 2025). The method improves output consistency by training the model to produce the same output or activation even when an input is semantically preserved but modified, based on the output actually sampled from the LLM for that input. The prior study uses a single output or activation as the ground truth. In contrast, we focus on the overall tendency rather than a single correct answer. We also differ in our primary focus, which is not on achieving consistency itself, but on evaluating the extent to which widely used LLMs possess consistency and what factors contribute to its improvement.

3 Method

We propose a method for measuring the belief consistency of the output tendency of an LLM through time. We define this consistency as the degree to which the transition of its output tendency remains logically consistent before and after a context extends with logical reasoning from the initial context. We expect that the context given to the LLM defines the state of the model at the time, and the extension of the context to be a state transition. To target a more general form of consistency, we focus on the potential tendency held by the model, specifically when its decision at that point is not explicitly stated in the prior context, and measure

this using sampling-based tendency measurement.

Figure 1 shows an overview of the proposed consistency measurement method. We give a task instruction to the LLM to perform an MCQA task, and its selection tendency is observed as a probability distribution based on the occurrence rate of each choice in the sampled output answers. Here we observe two types of selection tendencies: a prior distribution, where the answer frequencies are counted immediately after the task instruction, and a posterior distribution, where the model answers after a context involving logical reasoning is added to the same task instruction. We then quantitatively evaluate the logical consistency between the two tendencies using JS divergence, ultimately obtaining an consistency score. The range of the consistency score is $[0, 1]$. A higher score indicates that the model's output tendencies transition consistently with the contextual development.

3.1 Measuring Output Tendency via Sampling

The output tendency of an LLM is observed as the frequencies of decisions made over repeated generative sampling. Given context C, the LLM generates a subsequent token sequence by random sampling. From the generated token sequence, a decision d is extracted through a simple string matching. This generation process is iterated multiple times for the single context C, resulting in a set of decisions. Let N_d^C denote the number of times d is extracted from context C. We simply use the maximum likelihood estimation to estimate the probability over decisions:

$$p_d^C = \frac{N_d^C}{\sum_{d'} N_{d'}^C}. \tag{1}$$

In our experiment, we provide the task instruction and candidate options in a 3-choice MCQA format as the context and extract the model's answer as its decision. If multiple options match within a generated sequence, or if none are found, we treat it as a verbal error, and we also measure its probability p_{error}^C. Therefore, for a given context C, we obtain a probability distribution p^C composed of 4 classes: the options d_1, d_2, d_3 in the order they were presented in the prompt, and the verbal error ($p_{d_1}^C + p_{d_2}^C + p_{d_3}^C + p_{\text{error}}^C = 1$).

3.2 20 Questions-like MCQA Task

We have an LLM solve an MCQA task and observe its selection tendencies. However, typical MCQA tasks have a correct answer, which should introduce a strong bias into the tendencies. Therefore, inspired by the game of *20 Questions*, we design an MCQA task in which every provided option is a valid choice, thereby eliminating bias caused by answer correctness. We employ a conversational prompt throughout the experiment.

In this task, the LLM and the user play the roles of the questioner and the guesser, respectively. The LLM is asked to freely choose one entity, keeping its choice hidden until the user requests to reveal it. In the original *20 Questions*, the choice can be made without any restrictions. However, here we provide a fixed option set consisting of three entities to keep the observation computationally tractable. Since any option is valid in this task, there is no bias from a correct answer. This eliminates the need to consider task errors in the subsequent analysis, allowing us to focus purely on the transition of the thinking process.

Initially, the LLM takes a system prompt containing instructions to secretly select one of the given three entities. To expect the model to be consistent during performing the task, the system prompt also states that the model must answer subsequent Yes/No questions based on the selected entity, must not change the selected entity midway, and must comply with user requests for disclosure.

Under this task setting, we construct two types of conversation scenarios, namely the prior and the posterior scenarios, starting from the same system prompt with a certain option set. We expect that the context defines the state of the LLM, and that extending the context through subsequent logical inference induces a state transition. In the prior scenario, the LLM is prompted to reveal which option it has chosen immediately after receiving the system prompt. We generate the assistant response subsequently and treat it as the selected option by LLM. We determine which option is chosen by string matching with each option presented in the system prompt. The selection tendency observed at this point is regarded as the initial state of the LLM defined by the system prompt, which we refer to as the prior distribution. In the posterior scenario, starting from the same system prompt, we provide an additional conversation designed to elicit a logical update in the choice distribution, rather than prompting the LLM to reveal its choice immediately. According to the assumption above, this leads to a state transition, and the resulting tendency is referred to as the posterior distribution.

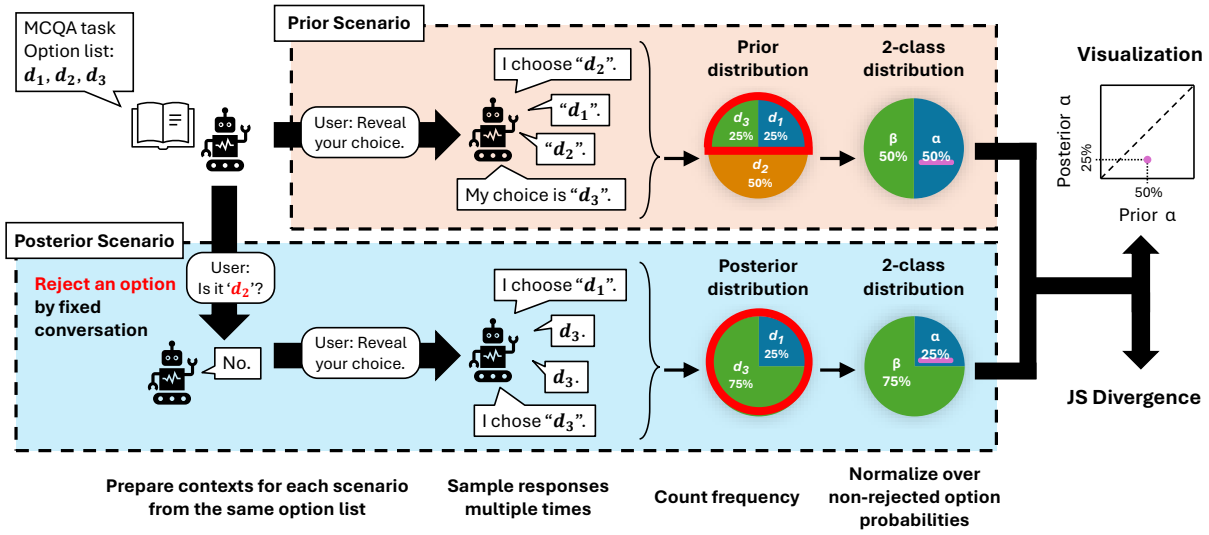


Figure 1: Overview of the experiment. Starting from the same MCQA system prompt, we construct two scenario contexts, a prior and a posterior, and evaluate consistency by comparing response tendencies. If an LLM's belief is consistently maintained, 2-class distributions in the two scenarios should match, resulting in a lower JS divergence.

The LLM is then prompted to reveal its chosen option as in the prior scenario.

We expect the selection tendency to transition consistently from the prior to the posterior according to the logical update. It is essential that the ideal posterior distribution is computationally predictable from the prior distribution so that the consistency can be evaluated. Moreover, we must ensure that the context does not include a concrete decision in any form before the model is explicitly asked to disclose it. Specifically, we construct a single user-assistant conversation turn that rejects the possibility of one of the three choices, narrowing the options down to two (e.g., "*User: Is it 'd_1'?*" → "*Assistant: No.*"). In this task, all choices are independent, and the rejection provides no additional information about the remaining two options. Therefore, the ideal selection ratio for these two options in the posterior scenario is expected to be the same as in the prior scenario. We used a simple dialogue for this rejection context. The underlying reason is as follows: if the rejection conversation contains complex logic, the model's reasoning ability may be overly challenged, leading to errors caused by failure to understand the logic. Such errors constitute noise that is unrelated to the purpose of evaluating consistency, and should therefore be avoided.

3.3 Two Class Consistency Score

In our task setting, the selection ratio between the two remaining valid options in the posterior scenario should be preserved from the ratio of those same two options in the prior scenario. Therefore, we define the consistency score by measuring how well this ratio is preserved.

Specifically, let α and β denote the two non-rejected options in the order they appear in the system prompt. We first normalize the selection probabilities over these two options for both the prior and posterior scenarios as follows:

$$\overline{p^C_{d'}} = \frac{p^C_{d'}}{p^C_\alpha + p^C_\beta}, \tag{2}$$

$$\overline{\boldsymbol{p}^C} = [\overline{p^C_\alpha}, \overline{p^C_\beta}], \tag{3}$$

where $d' \in \{\alpha, \beta\}$ and $C \in \{\text{prior}, \text{posterior}\}$.

We then compute the instance-level 2-class consistency score using the JS Divergence between the normalized 2-class distributions:

$$L^{\text{instance}} = 1 - \text{JSD}\big(\overline{\boldsymbol{p}^{\text{prior}}} \,||\, \overline{\boldsymbol{p}^{\text{posterior}}}\big). \tag{4}$$

Since we cannot assume either the prior or posterior better reflects the model's true tendency, we adopted JS divergence because it is symmetric.

To mitigate entity- and order-dependent biases, we evaluate multiple sets of three options and all permutations of each set. We additionally prepare two rejection templates to reduce prompt-specific biases. Each ordered option list combined with a rejection template constitutes one instance. The final 2-class consistency score is obtained by averaging across all instances. The consistency score

ranges from 0 to 1, with higher values indicating that decision-making transitions consistently follow contextual developments. A model exhibiting high consistency suggests that its internal reasoning is coherent, and the tokens it produces for thinking reflect a faithful track of its actual thinking process.

A model that has a strong preference to choose a specific option regardless of the context would also be evaluated as highly consistent under this metric. To separately quantify such biases, we additionally compute the instance-level entropy of the normalized 2-class distribution for each scenario and report their average over instances.

3.4 Three Class Consistency Score

In the 2-class consistency score above, the model's tendency toward the rejected option in the posterior context does not affect the score. This is because the selection of the rejected option pertains to logical consistency with explicitly mentioned content, which is different to our primary focus on the model's internal consistency. On the other hand, measuring how rarely the rejected option is selected is also valuable from the perspective of evaluating general logical reasoning ability. Therefore, we additionally compute the 3-class consistency score that accounts for the rejected option.

For the posterior distribution, we normalize the probabilities over all three options, including the rejected one. Meanwhile, for the prior distribution, we normalize only over the two appropriate options using the same procedure as in Equation (2), and assign a probability of 0% to the rejected option to represent the ideal posterior behavior. The final 3-class consistency score is then computed using the same formulation as Equation (4).

4 Experimental Setting

To analyze the effects of scaling, continued pre-training, and thinking processes, we experimented with various sizes of the following model families:

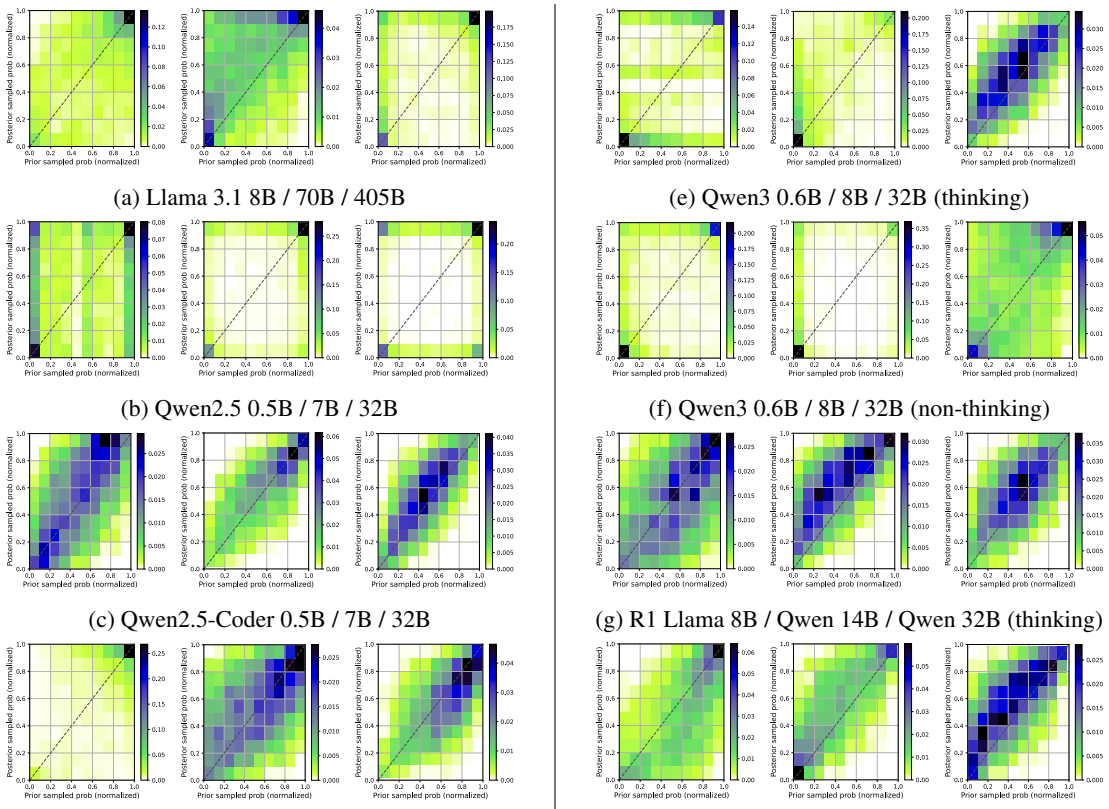
- Llama 3.1 family (8B, 70B, 405B) (Grattafiori et al., 2024)
- Qwen2.5 family (0.5B, 1.5B, 3B, 7B, 14B, 32B) (Yang et al., 2025b)
- Qwen2.5-Coder family (0.5B, 1.5B, 3B, 7B, 14B, 32B) (Hui et al., 2024)
- DeepSeek LLM and continual pre-trained models (DeepSeek LLM 7B, DeepSeek Coder 7B v1.5, DeepSeekMath 7B) (Bi et al., 2024)
- Qwen3 family (0.6B, 1.7B, 4B, 8B, 14B, 32B) (Yang et al., 2025a)
- DeepSeek-R1 distilled models (Llama 8B, Qwen 14B, Qwen 32B) (Guo et al., 2025)

All LLMs used are instruction-tuned models. Qwen2.5-Coder is a continual pre-trained model family on a coding corpus starting from Qwen2.5. Although Qwen2.5 has a 72B version, it is not available for Qwen2.5-Coder, so we omitted Qwen2.5 72B from this experiment. DeepSeek Coder v1.5 is a continual pre-trained model on a coding corpus to DeepSeek LLM, and DeepSeekMath is further continual pre-trained by adding mathematics text from DeepSeek Coder v1.5. Note that the continual pre-training was performed on the respective base models before instruction tuning, so the models experimented on here do not have a direct parent-child relationship. For the DeepSeek-R1 family, we limited our experiments to the distilled models due to the computation time constraint. The base model of DeepSeek-R1 Llama 8B is Llama-3.1 8B. The base models of DeepSeek-R1 Qwen 14B and 32B are Qwen2.5 14B and 32B, respectively. Contexts were constructed in a conversation format according to the provided chat templates. When using thinking models, experiments were conducted in both thinking and non thinking modes. Although DeepSeek-R1 is only trained in the thinking mode, we conducted a pseudo non-thinking mode experiment with a custom chat template imitating from the one used on Qwen3 and placing a closing *</think>* token immediately after the opening *<think>* token in the prompt. Other detailed experimental settings are described in Appendix A.

5 Results and Discussion

5.1 Factors Affecting the Consistency Score

Figure 2 shows the heatmaps of the plots of the instance-level 2-class normalized probabilities $\overline{p_\alpha^{\mathrm{C}}}$ for the prior and posterior cases, as defined in Equation 2, where each point represents a single instance and its probability before (x-axis) and after (y-axis) the added context. We show the original plots in Appendix C. If consistency holds, the points align along the $y = x$ line. Models with peaks in the bottom-right or top-left of the heatmap, like Qwen2.5 32B, signify a tendency for choosing the exact opposite option after a dialogue transition, even if they held a strong conviction for a specific option in the intermediate state. Table 1



(a) Llama 3.1 8B / 70B / 405B

(b) Qwen2.5 0.5B / 7B / 32B

(c) Qwen2.5-Coder 0.5B / 7B / 32B

(d) DeepSeek LLM / DeepSeek Coder / DeepSeekMath

(e) Qwen3 0.6B / 8B / 32B (thinking)

(f) Qwen3 0.6B / 8B / 32B (non-thinking)

(g) R1 Llama 8B / Qwen 14B / Qwen 32B (thinking)

(h) R1 Llama 8B / Qwen 14B / Qwen 32B (non-thinking)

Figure 2: Heatmaps of instance-level 2-class normalized probability plots. Each block shows three different-sized models from the same model family, except for the bottom-left block, which presents the heatmaps for DeepSeek LLM 7B and its successor models. The horizontal axis indicates the normalized prior probability for the first non-rejected option, $\overline{p_\alpha^{\text{prior}}}$, and the vertical axis indicates the normalized posterior probability, $\overline{p_\alpha^{\text{posterior}}}$, as defined in Equation 2. If consistency holds, the heatmap values concentrate along the $y = x$ line. Neither the Llama 3.1 family nor the Qwen2.5 family exhibits clear improvement with scaling. The Qwen2.5-Coder family shows increased entropy and a stronger convergence toward the $y = x$ line compared with the corresponding Qwen2.5 models. DeepSeek Coder and DeepSeekMath demonstrate more consistent behavior than the base DeepSeek LLM model. Qwen3 models display distinct behaviors depending on whether the thinking mode is enabled or disabled, while DeepSeek R1 models show more moderate differences. For the original plots, see Figures 4 and 5.

shows quantitative evaluation results based on our measurements. Indeed, for models where the visualized plots showed a clear difference, such as Qwen2.5 vs. Qwen2.5-Coder, or DeepSeek LLM vs. DeepSeek Coder and Math models, the plots are concentrated on the y=x line, confirming that the quantitatively evaluated consistency score reflects the intended concept. Among the models tested, the best consistency score (leftmost in the table) was achieved by Qwen2.5-Coder 14B in the 2-class setting, and Qwen3-32B (thinking) in the 3-class setting.

5.1.1 Effect of Model Scaling on General-domain Models

As seen in general-domain models like Llama3.1 and Qwen2.5, model size scaling alone does not naturally improve the belief consistency score. Larger models of these families have lower entropy, indicating a stronger tendency for bias among choices. The heatmaps also confirm that larger models have less distribution near the center.

5.1.2 Effect of Training on Math and Coding Text

As shown in the results of Qwen2.5 vs. Qwen2.5-Coder and the DeepSeek LLM vs its successor models in Figures 2 and Table 1, continual pre-training on mathematics and coding corpora strengthens the consistency, indicating that training on the corpora requiring the logical reasoning ability contributes to improving the consistency score. Interestingly, pre-training on such corpora also increases entropy, making the models less biased. For

Model	Think	Consistency		$p_{\text{invalid}}^{\text{posterior}}$	Entropy [bit]		Verbal Error	
		2 classes	3 classes		Prior	Posterior	Prior	Posterior
Llama 3.1 8B		0.9188	0.9048	2.50%	0.6323	0.6180	1.29%	1.87%
Llama 3.1 70B		0.8984	0.8833	2.82%	0.6868	0.6651	1.50%	0.70%
Llama 3.1 405B		0.8531	0.8531	**0.00%**	0.4065	0.4699	0.61%	0.42%
Qwen2.5 0.5B		0.7758	0.6684	14.11%	0.3741	0.6151	98.36%	29.31%
Qwen2.5 1.5B		0.9335	0.7636	27.37%	0.6052	0.6591	9.64%	0.89%
Qwen2.5 3B		0.7867	0.6882	12.64%	0.4649	0.2850	2.06%	1.56%
Qwen2.5 7B		0.8457	0.8382	1.43%	0.3754	0.3497	0.13%	0.42%
Qwen2.5 14B		0.7593	0.7590	0.04%	0.2693	0.1724	1.71%	1.74%
Qwen2.5 32B		0.7106	0.7106	0.00%	0.3508	0.1769	0.01%	0.09%
Qwen2.5-Coder 0.5B		0.9516	0.7505	6.59%	0.7992	0.7397	96.04%	56.01%
Qwen2.5-Coder 1.5B		0.9607	0.7571	27.86%	0.8075	0.7796	34.46%	5.81%
Qwen2.5-Coder 3B		0.9308	0.8057	18.61%	0.8414	0.7399	9.75%	3.55%
Qwen2.5-Coder 7B		0.9740	0.9023	12.97%	0.7685	0.7924	2.27%	2.33%
Qwen2.5-Coder 14B		**0.9856**	0.9071	13.02%	0.8271	0.8280	22.95%	6.54%
Qwen2.5-Coder 32B		0.9745	0.9051	12.05%	**0.8467**	0.8272	0.52%	0.23%
DeepSeek LLM 7B		0.9248	0.7806	19.15%	0.5525	0.4902	52.17%	14.82%
DeepSeek Coder 7B		0.9427	0.8763	9.62%	0.7432	0.7874	77.55%	22.45%
DeepSeekMath 7B		0.9693	0.8357	17.59%	0.7718	0.8159	13.13%	22.75%
Qwen3 0.6B	✓	0.8571	0.7634	3.37%	0.5736	0.2907	27.36%	86.19%
Qwen3 1.7B	✓	0.8533	0.8067	5.71%	0.5897	0.5583	7.07%	26.81%
Qwen3 4B	✓	0.9057	0.9021	0.73%	0.5857	0.5856	2.40%	1.77%
Qwen3 8B	✓	0.9135	0.9098	0.75%	0.4771	0.6034	2.04%	0.47%
Qwen3 14B	✓	0.9256	0.9243	0.26%	0.5689	0.7135	0.78%	0.37%
Qwen3 32B	✓	0.9659	**0.9545**	2.23%	0.8180	0.8473	2.85%	1.72%
Qwen3 0.6B		0.8829	0.6925	11.60%	0.4493	0.3734	6.98%	75.52%
Qwen3 1.7B		0.7944	0.7467	6.77%	0.0370	0.0525	0.33%	**0.08%**
Qwen3 4B		0.6993	0.6398	10.67%	0.2110	0.2290	1.14%	0.84%
Qwen3 8B		0.8290	0.7667	10.69%	0.2415	0.3994	1.08%	0.47%
Qwen3 14B		0.8313	0.8232	1.65%	0.4281	0.4302	**0.01%**	0.45%
Qwen3 32B		0.9127	0.9014	2.20%	0.7177	0.6265	33.06%	1.59%
DeepSeek R1 Llama 8B	✓	0.9354	0.8068	17.00%	0.7495	0.7626	54.46%	22.27%
DeepSeek R1 Qwen 14B	✓	0.9443	0.9023	7.56%	0.7704	0.7900	33.14%	8.26%
DeepSeek R1 Qwen 32B	✓	0.9665	0.9288	6.69%	0.8332	**0.8495**	13.52%	8.57%
DeepSeek R1 Llama 8B		0.9403	0.7619	22.05%	0.7254	0.7146	66.79%	19.66%
DeepSeek R1 Qwen 14B		0.9569	0.7962	24.81%	0.7210	0.7345	18.34%	6.88%
DeepSeek R1 Qwen 32B		0.9574	0.8654	16.00%	0.7795	0.8096	33.42%	4.66%

Table 1: Main results. All models are instruction-tuned. The best scores are highlighted in bold. "Consistency" represents the proposed consistency score. The "2 classes" and "3 classes" variants indicate whether the posterior probabilities are normalized only over the two valid options or over all three options, respectively. The column "$p_{\text{invalid}}^{\text{posterior}}$" reports the probability that the rejected option is still chosen in the posterior scenario. "Entropy" shows the average entropy of the normalized two-class probabilities for each scenario. "Verbal Error" denotes the probability that the model's response cannot be parsed.

example, peaks in the upper left and lower right corners shown in the heatmap for Qwen2.5, which represent the most inconsistent instances where the models completely switch their choices after the rejections, clearly disappear and shift to the y=x line in the one for Qwen2.5-Coder, specifically moving to the center region. These consistent trends confirm that math and coding training improves belief consistency while leading to less biased selections. Furthermore, scaling does not harm the consistency score in the Qwen2.5-Coder family, but instead leads to slight improvements. A possible reason for this result is the nature of mathematics and coding texts which involve many swappable blocks, thereby increasing the entropy, whereas in general text, changing the sentence order often leads to significant subsequent contextual changes. At the same time, LLMs still need to consistently manage how appropriate the appeared symbols are to reuse them at the later context appropriately, which should have a huge contribution for the consistency.

5.2 Effect of Thinking Mode

For Qwen3, enabling the thinking mode substantially improves the consistency score in all but the 0.6B variant. Even though the prior and posterior do not share any reasoning tokens, the thinking mode makes the final conclusions more consistent. In contrast, for the DeepSeek R1 family, enabling or disabling the thinking mode produces only minor

differences in the 2-class consistency score. Since the R1 models are not trained without the thinking mode, a direct comparison is inappropriate.

5.2.1 Logical and Verbalization Error Rates

We also report the verbalization error rates as well as the invalid probabilities in the posterior scenario $p_{\text{invalid}}^{\text{posterior}}$, which is the probabilities that the rejected option is chosen in the posterior scenario, in Table 1. For general-domain models such as Llama 3.1 and Qwen2.5, both verbalization errors and invalid probabilities decrease as model size increases. Since such explicit logical consistency should correlate with the abilities evaluated by typical benchmark tasks with correct answers, these results align with the general trend that scaling improves performance reported in the previous works (Grattafiori et al., 2024; Yang et al., 2025b). Interestingly, however, training on coding or mathematics tasks tends to worsen both the verbalization errors and invalid probabilities, even for larger models. Notably, while the original Qwen2.5 32B model produced almost no logical reasoning errors (4.44e-08%), Qwen2.5-Coder 32B generated more than 10% logical reasoning errors. The cause of this degradation is unclear, but given that Qwen2.5-Coder models of size 7B and larger converge to nearly the same invalid probability, it suggests that a non-negligible amount of inconsistent data may be included in the training corpus for this model family. One likely source is incomplete code snippets. Code fragments containing unimplemented blocks or typos may often occur in this domain.

See Appendix E for additional error analysis.

5.3 Problem Design

In our experiments, we treated higher entropy as preferable. This is because the model is completely free to choose any of the presented options under our prompt design, and the models we used were not trained to exhibit particular preferences. On the other hand, for models that have learned specific preferences, having certain tendencies should be appropriate, and the correct setting should be chosen depending on the intended application. Likewise, our treatment of consistency assumes that LLMs behave as naive models whose preference states transition in a logically coherent manner and our consistency score is defined based on this assumption. However, whether such logical transitions should occur can vary across model types. For example, we can consider the case applying the reinforcement-learning that a model is trained to select the option to which it previously assigned the highest probability, rather than updating its preferences through logical inference. Thus, what constitutes an appropriate level of consistency or entropy ultimately depends on the target application and the behavioral assumptions that it requires.

As suggested by the fact that the Qwen3 family exhibits variations in consistency scores depending on whether thinking mode is enabled, our evaluation method is inevitably influenced by the choice of prompts. In our experiments, we also tested different prompt templates for the rejection in the posterior scenario to demonstrate the prompt bias. Details are provided in Appendix C.1.3.

Designing contexts in which logical reasoning occurs must be done with care to avoid unintended state transition. When a reasoning process causes complex shifts in the original preference distribution, several issues arise: whether the model's pre-transition preferences are appropriate for performing the intended reasoning, what the post-transition probabilities should ideally look like, and whether the "reasoning" observed is in fact newly generated or merely a retrospective explanation of a conclusion already determined. These factors can introduce errors and biases that are qualitatively different from the logical transition we aim to evaluate. The context used in our experiments rejects only a single option and should not affect the relative preference between the remaining two options. Although this setup is simple, it avoids side effects on the valid two-choice comparison and allows a clear and well-defined computation of the expected post-transition probability distribution. In contrast, actual CoT or thinking tokens often involve tokens that trigger complex transitions.

6 Conclusion

We proposed belief consistency and an evaluation method for assessing the consistency of LLM beliefs across context extension. Experimental results showed that belief consistency is not naturally acquired through scaling, but that training on coding and math tasks is effective. We addressed the consistency in LLM tendencies that does not appear in the previous context; consistency with content that does appear explicitly is outside our scope. We believe that consistency breakdowns in such cases are a different type of problem, more related to instruction-following failures or insufficient LLM

training. In fact, in coding models, the consistency score improved while logical errors increased. Furthermore, in the evaluation of calibration against closed-set prediction probability ratios, we found that training on coding and mathematics corpora does not apply to that type of consistency.

Acknowledgments

This paper is based on results obtained from AIST policy-based budget project "R&D on Generative AI Foundation Models for the Physical Domain".

Limitations

We only evaluated up to one logical inference step. It is unknown if consistency is maintained in multi-step scenarios. For models with many failures, a large number of repeated samplings was required to obtain a stable score. To standardize the experimental settings for all models, we measured the choice ratio by performing 5000 samplings per context for non reasoning models. In our experimental setup, 10 contexts are generated per entity triple, resulting in a need for 50k generative samplings per entity triple. Although the context length is short, this is extremely costly. This is why we could not experiment on closed-weight models. To reduce computational load, we also attempted to calculate probabilities via search instead of sampling, but we left this strategy because it required an outrageous amount of memory, especially for tracking rarely chosen options.

We employed string matching for verbalization because of the manual analysis of generated responses, however, a higher-quality verbalization would likely yield a more accurate result.

References

Leonardo Bertolazzi, Davide Mazzaccara, Filippo Merlo, and Raffaella Bernardi. 2023. ChatGPT's information seeking strategy: Insights from the 20-questions game. In *Proceedings of the 16th International Natural Language Generation Conference*, pages 153–162, Prague, Czechia. Association for Computational Linguistics.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, and 67 others. 2024. DeepSeek LLM: Scaling open-source language models with longtermism. *Preprint*, arXiv:2401.02954.

Fengxiang Cheng, Haoxuan Li, Fenrong Liu, Robert van Rooij, Kun Zhang, and Zhouchen Lin. 2025. Empowering LLMs with logical reasoning: A comprehensive survey. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 10400–10408. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Zorik Gekhman, Eyal Ben-David, Hadas Orgad, Eran Ofek, Yonatan Belinkov, Idan Szpektor, Jonathan Herzig, and Roi Reichart. 2025. Inside-out: Hidden factual knowledge in LLMs. In *Second Conference on Language Modeling*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, and 5 others. 2024. Qwen2.5-Coder technical report. *Preprint*, arXiv:2409.12186.

Alex Irpan, Alexander Matt Turner, Mark Kurzeja, David K. Elson, and Rohin Shah. 2025. Consistency training helps stop sycophancy and jailbreaks. *Preprint*, arXiv:2510.27062.

J. Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.

Dasha Metropolitansky and Jonathan Larson. 2025. Towards effective extraction and evaluation of factual claims. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6996–7045, Vienna, Austria. Association for Computational Linguistics.

Masanari Ohi, Masahiro Kaneko, Ryuto Koike, Mengsay Loem, and Naoaki Okazaki. 2024. Likelihood-based mitigation of evaluation bias in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3237–3245, Bangkok, Thailand. Association for Computational Linguistics.

Pouya Pezeshkpour and Estevam Hruschka. 2024. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017, Mexico City, Mexico. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025b. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Yizhe Zhang, Jiarui Lu, and Navdeep Jaitly. 2024. Probing the multi-turn planning capabilities of LLMs via 20 question games. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1495–1516, Bangkok, Thailand. Association for Computational Linguistics.

A Detailed Experimental Settings

We limit the maximum sequence length of the generated sequence and terminate the generation when the length of the generated sequence exceeds it even if an end-of-sequence token has not appeared.

Although some models used in the experiment were not trained on prompts including system prompts, making their use non-recommended, we set the prompt roles with the same configuration for all models this time. When in the thinking mode of a reasoning model, the thinking process is used only for measuring the selection tendency of each context; it is not included in the dialogue leading up to that point. Therefore, the thinking process conducted in the prior is not reflected in the posterior measurement. In this study, we also experimented with DeepSeek R1 models. The original R1 family does not have a non-thinking mode and was not trained for it. However, referencing Qwen3, we prepared a chat template that simulates a no-thinking mode by placing a </think> tag immediately after the <think> tag and conducted the experiment.

The number of samples per context was 50 in thinking mode and 5000 otherwise. Cases where neither of the two appropriate options in the posterior context was sampled even once (either during the prior or posterior measurement) were excluded from the evaluation, as Equation 2 could not be calculated. The maximum number of generated tokens for each sampling was 3000 for reasoning models and 100 for other models. If this limit was exceeded without an end-of-sequence token appearing, generation was terminated. During sampling, we used unmodified logits set to temperature 1, and no penalties such as length penalty were applied.

Figure 3 shows the prompt templates used in our experiments. To evaluate the influence of prompt bias in the dialogue patterns, we tested two types: a dialogue path that presents one option and denies it (User: "*is it 'X'?*" -> Assistant: "*No.*"), and a dialogue path that presents two options and confirms them (User: "*Is it either 'X' or 'Y'?*" -> Assistant: "*Yes.*"). To mitigate the effect of position bias, we prepared the all three rejection variations for the former template and all six permutation patterns for the latter template as separate instances. Therefore, for one set of three entities, we conducted the set number of sampling iterations separately for one prior context and nine posterior contexts, totaling 10 contexts.

Wikipedia section titles were used as entity candidates for the options. We sorted the collected Wikipedia section titles based on likelihood calculated by Llama 3.1 8B, adopted the top 1000 as plausible entities, and a randomly selected set of 3 entities from this pool constituted one context C. To evaluate entity bias, we also conducted experiments using a setting where three entity sets were chosen from the 1000 items ranked 10,001st to 11,000th. Due to experimental time constraints, experiments with the latter entity candidates were not conducted on the reasoning models. We extracted 500 ordered combinations of 3 entities from each entity candidate set for the experiment. These 500 sets were identical across all model experiments.

System prompt

```
You will see 3 options. As the first step of your task, you must secretly choose one of
them and keep in your mind.
Your main task is answering the given questions consistently based on your choice.
You cannot change your choice during the answering process.
You must not directly mention your chosen word except when the user requests 'Reveal your
choice.'
Questions are Yes-No questions. You must answer 'Yes.' or 'No.'

OPTIONS:
- <OPTION d1>
- <OPTION d2>
- <OPTION d3>
```

User prompt for sampling the choice

```
Reveal your choice.
```

User and assistant prompts for a posterior scenario: presenting one option and denying it

User: Is it '<OPTION X>'?
Assistant: No.

User and assistant prompts for a posterior scenario: presenting two options and confirming them

User: Is it either '<OPTION X>' or '<OPTION Y>'?
Assistant: Yes.

Figure 3: Prompt templates used in our experiments. "<OPTION>" denotes a placeholder for an actual option. We give the task description as a system prompt. In the prior scenario, a user prompt requesting disclosure of the choice is given immediately after the system prompt. In the posterior scenario, the candidate options are first narrowed down to two using one of two prompt patterns, after which a user prompt requesting disclosure of the choice is given. To cover all possible option combinations and orderings in the posterior scenario, we used three instantiations for the case in which one option is presented and denied, and six instantiations for the case in which two options are presented and confirmed.

The selected entity was determined by string matching. Specifically, the generated response was converted to lowercase, and a match was performed based on whether the lowercased entity appeared in the response. If no entity was found in the generated response, or if multiple entities matched, it was treated as a verbal error. For parsing decisions in thinking mode, string matching was performed on the text appearing after the </think> tag. Generation cases where the </think> tag did not appear were also counted as verbal errors.

We used the January 2025 dump of Wikipedia to extract entity candidates. We used the Python package *vLLM*[1] (*v0.7.3*) together with *PyTorch*[2] (*v2.5.1+cu124*) and *Transformers*[3] (*v4.51.3*) for our experiments.

[1]https://github.com/vllm-project/vllm
[2]https://pytorch.org/
[3]https://github.com/huggingface/transformers

B Consideration of Other Evaluation Metrics

In addition to the proposed formulation of the consistency score, we also evaluated alternative scoring schemes using mean squared error (MSE) and mean absolute error (MAE) in place of the JS divergence. The results are summarized in Table 2. As shown in the table, all scoring methods exhibit similar overall trends. Since the JS divergence provides a more natural interpretation in terms of probability distributions, we employ it as the final scoring method in our study.

We also conducted an evaluation targeting scenarios where the selected option changes deterministically. Table 3 shows the average value of the risk of selection switching, defined as follows:
1. We define a score, *Switch*, which evaluates the risk of a hard switch in choices between the prior and posterior contexts when sampling only once

Model	Think	$1 - \text{JS\_Divergence}$	MSE	MAE
Llama 3.1 8B		0.9188	0.0868	0.2183
Llama 3.1 70B		0.8984	0.1097	0.2551
Llama 3.1 405B		0.8531	0.1487	0.2674
Qwen2.5 0.5B		0.7758	0.2212	0.3682
Qwen2.5 1.5B		0.9335	0.0715	0.1973
Qwen2.5 3B		0.7867	0.2122	0.3300
Qwen2.5 7B		0.8457	0.1530	0.2594
Qwen2.5 14B		0.7593	0.2384	0.3163
Qwen2.5 32B		0.7106	0.2874	0.3789
Qwen2.5-Coder 0.5B		0.9516	0.0528	0.1814
Qwen2.5-Coder 1.5B		0.9607	0.0452	0.1646
Qwen2.5-Coder 3B		0.9308	0.0806	0.2192
Qwen2.5-Coder 7B		0.9740	0.0300	0.1317
Qwen2.5-Coder 14B		**0.9856**	**0.0167**	**0.1003**
Qwen2.5-Coder 32B		0.9745	0.0299	0.1356
DeepSeek LLM 7B		0.9248	0.0748	0.1915
DeepSeek Coder 7B		0.9427	0.0640	0.1969
DeepSeekMath 7B		0.9693	0.0353	0.1457
Qwen3 0.6B	✓	0.8571	0.1229	0.2521
Qwen3 1.7B	✓	0.8533	0.1498	0.3005
Qwen3 4B	✓	0.9057	0.0938	0.2269
Qwen3 8B	✓	0.9135	0.0820	0.2085
Qwen3 14B	✓	0.9256	0.0727	0.2074
Qwen3 32B	✓	0.9659	0.0391	0.1575
Qwen3 0.6B		0.8829	0.1162	0.2244
Qwen3 1.7B		0.7944	0.2045	0.2226
Qwen3 4B		0.6993	0.3002	0.3786
Qwen3 8B		0.8290	0.1686	0.2703
Qwen3 14B		0.8313	0.1710	0.2935
Qwen3 32B		0.9127	0.0946	0.2348
DeepSeek R1 Llama 8B	✓	0.9354	0.0688	0.2049
DeepSeek R1 Qwen 14B	✓	0.9443	0.0617	0.1991
DeepSeek R1 Qwen 32B	✓	0.9665	0.0388	0.1563
DeepSeek R1 Llama 8B		0.9403	0.0665	0.1970
DeepSeek R1 Qwen 14B		0.9569	0.0479	0.1648
DeepSeek R1 Qwen 32B		0.9574	0.0491	0.1753

Table 2: Comparison with other candidate consistency score evaluation functions

Model	Think	Free-form			Closed		
		Switch	HSwitch	Entropy	Switch	HSwitch	Entropy
Llama 3.1 8B		36.8%	15.1%	0.6323	29.6%	17.2%	0.4049
Llama 3.1 70B		41.9%	18.0%	0.6868	40.2%	20.1%	0.6081
Llama 3.1 405B		34.0%	21.6%	0.4065	32.1%	21.2%	0.3684
Qwen2.5 0.5B		44.6%	32.1%	0.3741	44.0%	29.8%	0.4550
Qwen2.5 1.5B		35.7%	14.2%	0.6052	37.2%	16.9%	0.5999
Qwen2.5 3B		37.6%	25.1%	0.4649	34.4%	28.6%	0.2250
Qwen2.5 7B		31.1%	20.4%	0.3754	32.7%	27.0%	0.2229
Qwen2.5 14B		33.3%	26.7%	0.2693	36.2%	31.5%	0.2070
Qwen2.5 32B		40.1%	31.4%	0.3508	38.6%	32.2%	0.2625
Qwen2.5-Coder 0.5B		41.2%	10.9%	0.7992	37.7%	19.2%	0.5353
Qwen2.5-Coder 1.5B		41.8%	10.1%	0.8075	38.9%	10.4%	0.7347
Qwen2.5-Coder 3B		45.1%	13.5%	0.8414	42.5%	15.7%	**0.7453**
Qwen2.5-Coder 7B		39.4%	8.4%	0.7685	36.5%	11.4%	0.6496
Qwen2.5-Coder 14B		40.8%	**5.9%**	0.8271	37.0%	**7.8%**	0.7404
Qwen2.5-Coder 32B		42.8%	8.0%	**0.8467**	30.1%	17.5%	0.4061
DeepSeek LLM 7B		30.5%	12.6%	0.5525	35.7%	29.7%	0.2174
DeepSeek Coder 7B		42.1%	13.6%	0.7432	30.3%	18.4%	0.3919
DeepSeekMath 7B		40.7%	9.7%	0.7718	33.1%	12.9%	0.5523
Qwen3 0.6B	✓	32.2%	15.6%	0.5736	29.9%	19.2%	0.3705
Qwen3 1.7B	✓	41.1%	22.1%	0.5897	26.5%	24.3%	0.0926
Qwen3 4B	✓	35.9%	16.0%	0.5857	31.6%	27.6%	0.1614
Qwen3 8B	✓	32.6%	16.0%	0.4771	30.0%	24.2%	0.2127
Qwen3 14B	✓	36.6%	15.8%	0.5689	39.4%	30.8%	0.3091
Qwen3 32B	✓	43.4%	10.2%	0.8180	36.0%	19.3%	0.5298
Qwen3 0.6B		29.6%	16.2%	0.4493	29.9%	19.2%	0.3735
Qwen3 1.7B		**22.3%**	21.5%	0.0370	**23.1%**	22.3%	0.0345
Qwen3 4B		39.4%	34.1%	0.2110	31.4%	27.5%	0.1599
Qwen3 8B		30.7%	23.7%	0.2415	29.9%	24.3%	0.2076
Qwen3 14B		35.9%	23.2%	0.4281	39.3%	30.7%	0.3092
Qwen3 32B		40.1%	15.3%	0.7177	36.0%	19.4%	0.5278
DeepSeek R1 Llama 8B	✓	42.3%	13.8%	0.7495	29.2%	18.0%	0.3933
DeepSeek R1 Qwen 14B	✓	42.8%	13.3%	0.7704	32.1%	14.9%	0.5188
DeepSeek R1 Qwen 32B	✓	43.9%	10.0%	0.8332	39.8%	15.5%	0.6595
DeepSeek R1 Llama 8B		39.8%	13.0%	0.7254	29.2%	18.0%	0.3933
DeepSeek R1 Qwen 14B		38.4%	10.8%	0.7210	32.1%	14.9%	0.5188
DeepSeek R1 Qwen 32B		42.2%	11.6%	0.7795	39.8%	15.5%	0.6595

Table 3: Average probability of the selection switching from prior to posterior. HSwitch uses a hinge-type score that tolerates selection switches that can occur even when the prior and posterior probabilities are identical.

per instance.

$$\text{Switch} := \mathbb{E}_{\text{instance}}\left[p_\alpha^{\text{prior}}\,\overline{p_\beta^{\text{posterior}}} + \overline{p_\beta^{\text{prior}}}\,p_\alpha^{\text{posterior}}\right]. \quad (5)$$

2. Models with high entropy often switch even in the ideal situation where the posterior and prior distributions are identical. Therefore, a simple Switch risk evaluation can be overly punitive. We also implemented HSwitch, which discounts the selection switches that occur even in the ideal case:

$$\text{HSwitch} := \mathbb{E}_{\text{instance}}\Big[p_\alpha^{\text{prior}} \cdot \max\Big(\overline{p_\beta^{\text{posterior}}} - \overline{p_\beta^{\text{prior}}},\ 0\Big) + \overline{p_\beta^{\text{prior}}} \cdot \max\Big(\overline{p_\alpha^{\text{posterior}}} - \overline{p_\alpha^{\text{prior}}},\ 0\Big)\Big]. \quad (6)$$

When comparing HSwitch, the risk of switching is lower in free-form generation than in the closed set in most cases. On the other hand, when comparing Switch, the closed set often has a lower risk, but this is not a strong trend.

C Detailed Analysis

Figures 4 and 5 show the plots of the normalized probability comparison between the prior and the posterior scenario for each instance. The heatmaps are the same as shown in Figure 2.

C.1 Evaluation of Bias

C.1.1 Likelihood Bias

To examine the impact of candidate entity likelihood on consistency, we evaluated the consistency scores when sampling entity sets with different likelihood levels. The results are shown in Table 4. We find that when the provided options have similar likelihoods, the absolute likelihood level itself has little effect on the consistency score.

C.1.2 Position Bias

Regardless of the presence of continual pre-training on coding and mathematics texts, we frequently observed peaks concentrating in the upper-right area of the heatmaps in Figure 2. The probability plotted in these figures is always the normalized selection rate for the first-appearing valid option. The prevalence of high values indicates that all models tend to select the entity that appears first, reflecting the influence of position bias confirmed in prior research. Position bias appears stronger than entity type bias.

Model	Consistency (2-class)	
	Entity Likelihood Top-N	
	1-1000	10001-11000
Llama 3.1 8B	0.9188	0.9199
Llama 3.1 70B	0.8984	0.9068
Llama 3.1 405B	0.8531	0.8724
Qwen2.5 0.5B	0.7758	0.7736
Qwen2.5 1.5B	0.9335	0.9288
Qwen2.5 3B	0.7867	0.7824
Qwen2.5 7B	0.8457	0.8568
Qwen2.5 14B	0.7593	0.7412
Qwen2.5 32B	0.7106	0.7351
Qwen2.5-Coder 0.5B	0.9516	0.9498
Qwen2.5-Coder 1.5B	0.9607	0.9593
Qwen2.5-Coder 3B	0.9308	0.9333
Qwen2.5-Coder 7B	0.9740	0.9737
Qwen2.5-Coder 14B	**0.9856**	**0.9875**
Qwen2.5-Coder 32B	0.9745	0.9742
DeepSeek LLM 7B	0.9248	0.9287
DeepSeek Coder 7B	0.9427	0.9468
DeepSeekMath 7B	0.9693	0.9696

Table 4: Consistency scores for entity candidates with different likelihoods.

In some models, the plot is skewed to the upper-left of the y=x line. Unlike the quantitative evaluation, this figure plots only the probability for the option that appeared first in the system prompt, among the two appropriate posterior options. Therefore, models with this tendency select the first-appearing entity more often in the posterior than in the prior, suggesting that position bias may strengthen when the number of options is reduced.

Table 5 shows the average normalized probabilities for the first non-rejected option in the posterior scenario. The farther the probability is from 0.5, the stronger the position bias. Although the table shows clear evidence of position bias, the large standard deviations make the results difficult to interpret from the table alone. We consider that Figure 2 provides more direct insight by showing the full distributional patterns, whereas the table offers a complementary quantitative summary.

C.1.3 Prompt Bias

Tables 6 and 7 show the consistency scores for each rejection template. Overall, rejection using the "*Is it X?*" template yields higher consistency than the "*Is it either X or Y?*" template. This result suggests that simpler prompts tend to better preserve consistency.

D Calibration

We examine the consistency between the normalized probabilities obtained from free-form sampled generations and the log-probabilities of the closed-

Model	**Think**	$\overline{p_\alpha^{\text{prior}}}$	$\overline{p_\alpha^{\text{posterior}}}$
Llama 3.1 8B		0.6265 ± 0.3019	0.6302 ± 0.3070
Llama 3.1 70B		0.4419 ± 0.2991	0.6040 ± 0.2950
Llama 3.1 405B		0.4869 ± 0.4018	0.6321 ± 0.3594
Qwen2.5 0.5B		0.4572 ± 0.4020	0.5390 ± 0.3316
Qwen2.5 1.5B		0.7028 ± 0.2711	0.6074 ± 0.2980
Qwen2.5 3B		0.6652 ± 0.3468	0.5571 ± 0.4304
Qwen2.5 7B		0.6048 ± 0.3970	0.6364 ± 0.3937
Qwen2.5 14B		0.5475 ± 0.4359	0.5955 ± 0.4520
Qwen2.5 32B		0.5488 ± 0.4149	0.6157 ± 0.4459
Qwen2.5-Coder 0.5B		0.4875 ± 0.2492	0.5365 ± 0.2782
Qwen2.5-Coder 1.5B		0.5741 ± 0.2329	0.5023 ± 0.2598
Qwen2.5-Coder 3B		0.6056 ± 0.1976	0.5158 ± 0.2816
Qwen2.5-Coder 7B		0.5859 ± 0.2528	0.6131 ± 0.2276
Qwen2.5-Coder 14B		0.6172 ± 0.2018	0.6334 ± 0.1915
Qwen2.5-Coder 32B		0.4652 ± 0.2169	0.5407 ± 0.2289
DeepSeek LLM 7B		0.7243 ± 0.2780	0.7357 ± 0.2950
DeepSeek Coder 7B		0.5337 ± 0.2760	0.5543 ± 0.2499
DeepSeekMath 7B		0.6409 ± 0.2247	0.5993 ± 0.2186
Qwen3 0.6B	✓	0.4486 ± 0.3412	0.4627 ± 0.4245
Qwen3 1.7B	✓	0.4667 ± 0.3377	0.5665 ± 0.3459
Qwen3 4B	✓	0.4086 ± 0.3296	0.4574 ± 0.3401
Qwen3 8B	✓	0.2841 ± 0.3105	0.4005 ± 0.3211
Qwen3 14B	✓	0.3563 ± 0.3174	0.4473 ± 0.2875
Qwen3 32B	✓	0.4643 ± 0.2350	0.5479 ± 0.2144
Qwen3 0.6B		0.4127 ± 0.3801	0.5251 ± 0.4095
Qwen3 1.7B		0.9035 ± 0.2819	0.7421 ± 0.4247
Qwen3 4B		0.5809 ± 0.4455	0.5864 ± 0.4402
Qwen3 8B		0.2234 ± 0.3492	0.3750 ± 0.3841
Qwen3 14B		0.4262 ± 0.3884	0.5148 ± 0.3947
Qwen3 32B		0.4941 ± 0.2914	0.5539 ± 0.3253
DeepSeek R1 Llama 8B	✓	0.5595 ± 0.2660	0.5258 ± 0.2664
DeepSeek R1 Qwen 14B	✓	0.4477 ± 0.2576	0.6098 ± 0.2291
DeepSeek R1 Qwen 32B	✓	0.4950 ± 0.2275	0.5536 ± 0.2113
DeepSeek R1 Llama 8B		0.6146 ± 0.2635	0.5249 ± 0.2916
DeepSeek R1 Qwen 14B		0.4176 ± 0.2776	0.5154 ± 0.2828
DeepSeek R1 Qwen 32B		0.4356 ± 0.2525	0.5582 ± 0.2366

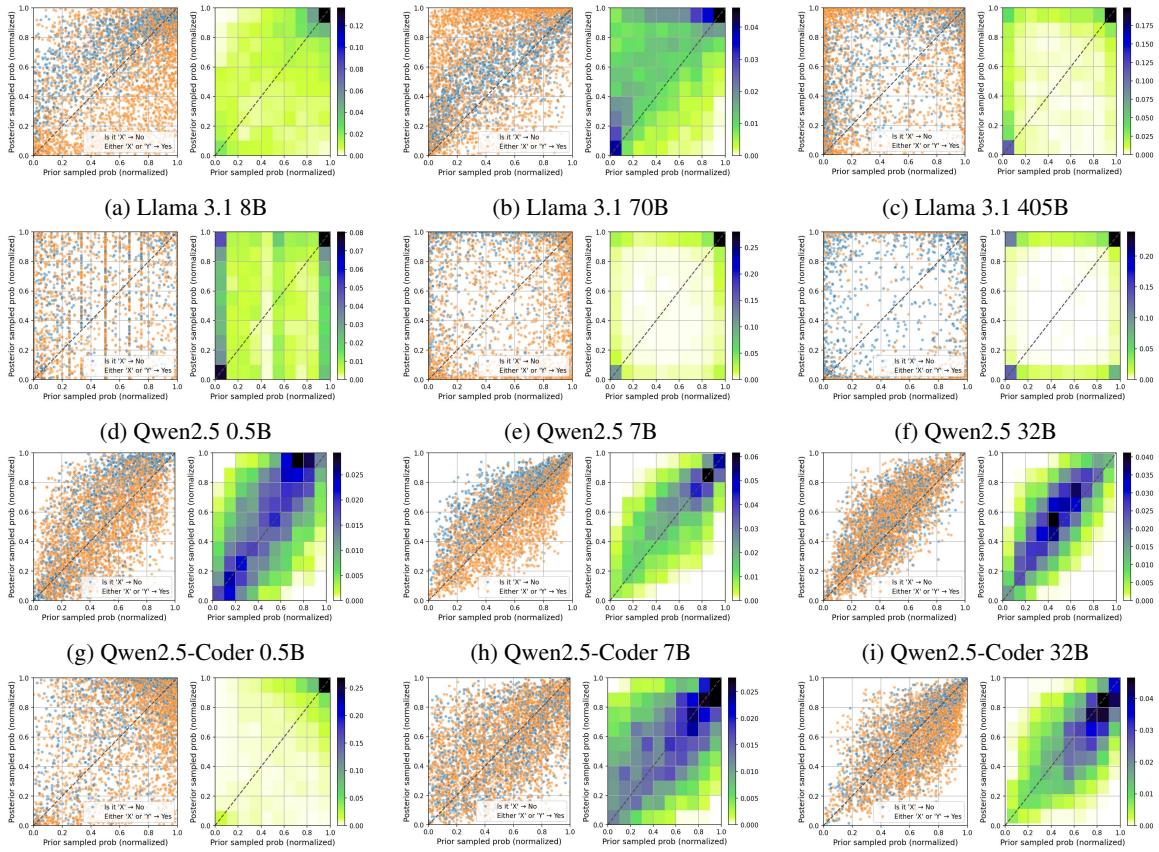
Table 5: Average normalized probabilities for the first non-rejected option in the posterior scenario. We report the mean probabilities and their standard deviations. Probabilities that differ substantially from 0.5 indicate a stronger position bias.

Model	Think	Consistency		$p_{\text{invalid}}^{\text{posterior}}$	Entropy [bit]		Verbal Error	
		2 classes	3 classes		Prior	Posterior	Prior	Posterior
Llama 3.1 8B		0.9467	0.9145	5.91%	0.6323	0.6062	1.29%	2.41%
Llama 3.1 70B		0.9561	0.9114	8.28%	0.6868	0.7839	1.50%	0.99%
Llama 3.1 405B		0.9211	0.9211	**0.00%**	0.4065	0.5092	0.61%	0.41%
Qwen2.5 0.5B		0.7912	0.6677	10.91%	0.3741	0.6337	98.36%	50.35%
Qwen2.5 1.5B		0.9587	0.7339	34.97%	0.6052	0.5989	9.64%	0.84%
Qwen2.5 3B		0.8282	0.1381	88.08%	0.4442	0.1474	2.34%	1.96%
Qwen2.5 7B		0.8403	0.8384	0.40%	0.3754	0.2550	0.13%	0.79%
Qwen2.5 14B		0.8127	0.8118	0.13%	0.2695	0.1960	1.65%	1.98%
Qwen2.5 32B		0.8426	0.8426	0.00%	0.3508	0.3368	0.01%	0.26%
Qwen2.5-Coder 0.5B		0.9520	0.4043	14.16%	0.7992	0.6667	96.04%	80.92%
Qwen2.5-Coder 1.5B		0.9766	0.4778	63.75%	0.8075	0.7333	34.46%	6.44%
Qwen2.5-Coder 3B		0.9815	0.6418	48.59%	0.8414	0.7751	9.75%	5.57%
Qwen2.5-Coder 7B		0.9708	0.8554	20.44%	0.7685	0.7534	2.27%	2.23%
Qwen2.5-Coder 14B		**0.9844**	0.9666	3.15%	0.8271	**0.8545**	22.95%	7.56%
Qwen2.5-Coder 32B		0.9750	**0.9711**	0.78%	**0.8467**	0.8320	0.52%	0.26%
DeepSeek LLM 7B		0.9353	0.7907	20.79%	0.5525	0.4208	52.17%	5.93%
DeepSeek Coder 7B		0.9465	0.7820	24.73%	0.7432	0.7840	77.55%	11.72%
DeepSeekMath 7B		0.9781	0.6697	39.98%	0.7718	0.7873	13.13%	17.08%
Qwen3 0.6B	✓	0.8531	0.5593	10.84%	0.5828	0.2844	26.69%	79.97%
Qwen3 1.7B	✓	0.8912	0.7709	13.66%	0.5900	0.6053	7.07%	34.32%
Qwen3 4B	✓	0.9007	0.9002	0.08%	0.5857	0.5646	2.40%	0.60%
Qwen3 8B	✓	0.9127	0.9116	0.15%	0.4771	0.5735	2.04%	0.55%
Qwen3 14B	✓	0.9233	0.9231	0.04%	0.5689	0.7159	0.78%	0.75%
Qwen3 32B	✓	0.9693	0.9678	0.31%	0.8180	0.8429	2.85%	1.55%
Qwen3 0.6B		0.9084	0.4509	33.66%	0.4501	0.3792	6.99%	45.16%
Qwen3 1.7B		0.8661	0.7819	11.54%	0.0379	0.0463	0.33%	**0.20%**
Qwen3 4B		0.6976	0.5585	24.00%	0.2110	0.2723	1.19%	0.75%
Qwen3 8B		0.8526	0.7924	10.38%	0.2413	0.4582	1.08%	0.66%
Qwen3 14B		0.8352	0.8347	0.02%	0.4281	0.4782	**0.01%**	1.18%
Qwen3 32B		0.9321	0.9262	1.20%	0.7177	0.6593	33.06%	1.49%
DeepSeek R1 Llama 8B	✓	0.9511	0.6825	32.72%	0.7495	0.7605	54.46%	26.22%
DeepSeek R1 Qwen 14B	✓	0.9461	0.9104	6.41%	0.7704	0.7976	33.14%	8.15%
DeepSeek R1 Qwen 32B	✓	0.9634	0.9373	4.67%	0.8332	0.8341	13.52%	8.41%
DeepSeek R1 Llama 8B		0.9737	0.5730	45.46%	0.7254	0.6621	66.79%	21.89%
DeepSeek R1 Qwen 14B		0.9651	0.7324	34.37%	0.7210	0.7249	18.34%	6.22%
DeepSeek R1 Qwen 32B		0.9512	0.8700	13.97%	0.7795	0.8080	33.42%	6.53%

Table 6: Evaluation of prompt bias. Posterior context is “Is it X?” -> “No.” only.

Model	Think	Consistency		$p_{\text{invalid}}^{\text{posterior}}$	Entropy [bit]		Verbal Error	
		2 classes	3 classes		Prior	Posterior	Prior	Posterior
Llama 3.1 8B		0.9048	0.8999	0.80%	0.6323	0.6239	1.29%	1.60%
Llama 3.1 70B		0.8696	0.8692	0.08%	0.6868	0.6056	1.50%	0.55%
Llama 3.1 405B		0.8191	0.8191	**0.00%**	0.4065	0.4502	0.61%	0.43%
Qwen2.5 0.5B		0.7681	0.6688	15.70%	0.3741	0.6058	98.36%	18.80%
Qwen2.5 1.5B		0.9209	0.7785	23.57%	0.6052	0.6892	9.64%	0.92%
Qwen2.5 3B		0.7800	0.7770	0.45%	0.4682	0.3072	2.02%	1.50%
Qwen2.5 7B		0.8484	0.8381	1.94%	0.3754	0.3970	0.13%	0.24%
Qwen2.5 14B		0.7327	0.7327	0.00%	0.2692	0.1606	1.74%	1.62%
Qwen2.5 32B		0.6447	0.6447	0.00%	0.3508	0.0970	0.01%	**0.00%**
Qwen2.5-Coder 0.5B		0.9514	0.9235	2.81%	0.7992	0.7762	96.04%	43.56%
Qwen2.5-Coder 1.5B		0.9527	0.8968	9.91%	0.8075	0.8028	34.46%	5.50%
Qwen2.5-Coder 3B		0.9055	0.8877	3.61%	0.8414	0.7224	9.75%	2.55%
Qwen2.5-Coder 7B		0.9756	0.9258	9.24%	0.7685	0.8119	2.27%	2.37%
Qwen2.5-Coder 14B		**0.9862**	0.8773	17.96%	0.8271	0.8147	22.95%	6.03%
Qwen2.5-Coder 32B		0.9742	0.8721	17.69%	**0.8467**	0.8249	0.52%	0.21%
DeepSeek LLM 7B		0.9195	0.7756	18.33%	0.5525	0.5250	52.17%	19.26%
DeepSeek Coder 7B		0.9408	0.9235	2.07%	0.7432	0.7890	77.55%	27.81%
DeepSeekMath 7B		0.9650	0.9187	6.40%	0.7718	0.8303	13.13%	25.58%
Qwen3 0.6B	✓	0.8587	0.8434	0.45%	0.5701	0.2932	27.62%	88.63%
Qwen3 1.7B	✓	0.8344	0.8246	1.74%	0.5895	0.5348	7.07%	23.06%
Qwen3 4B	✓	0.9083	0.9030	1.06%	0.5857	0.5961	2.40%	2.36%
Qwen3 8B	✓	0.9140	0.9089	1.05%	0.4771	0.6183	2.04%	0.43%
Qwen3 14B	✓	0.9268	0.9250	0.37%	0.5689	0.7123	0.78%	0.17%
Qwen3 32B	✓	0.9642	**0.9479**	3.19%	0.8180	0.8495	2.85%	1.81%
Qwen3 0.6B		0.8702	0.8119	0.69%	0.4489	0.3705	6.97%	90.52%
Qwen3 1.7B		0.7610	0.7303	4.54%	0.0365	0.0553	0.33%	0.03%
Qwen3 4B		0.7001	0.6803	4.03%	0.2110	0.2074	1.12%	0.88%
Qwen3 8B		0.8173	0.7539	10.84%	0.2416	0.3700	1.08%	0.37%
Qwen3 14B		0.8294	0.8175	2.46%	0.4281	0.4063	**0.01%**	0.09%
Qwen3 32B		0.9031	0.8891	2.70%	0.7177	0.6100	33.06%	1.65%
DeepSeek R1 Llama 8B	✓	0.9275	0.8690	9.14%	0.7495	0.7637	54.46%	20.30%
DeepSeek R1 Qwen 14B	✓	0.9435	0.8983	8.14%	0.7704	0.7863	33.14%	8.31%
DeepSeek R1 Qwen 32B	✓	0.9680	0.9246	7.69%	0.8332	**0.8572**	13.52%	8.65%
DeepSeek R1 Llama 8B		0.9237	0.8564	10.35%	0.7254	0.7409	66.79%	18.54%
DeepSeek R1 Qwen 14B		0.9527	0.8281	20.03%	0.7210	0.7393	18.34%	7.21%
DeepSeek R1 Qwen 32B		0.9605	0.8631	17.01%	0.7795	0.8105	33.42%	3.73%

Table 7: Evaluation of prompt bias. Posterior context is “Is it either X or Y?” -> “Yes.” only.



(a) Llama 3.1 8B (b) Llama 3.1 70B (c) Llama 3.1 405B

(d) Qwen2.5 0.5B (e) Qwen2.5 7B (f) Qwen2.5 32B

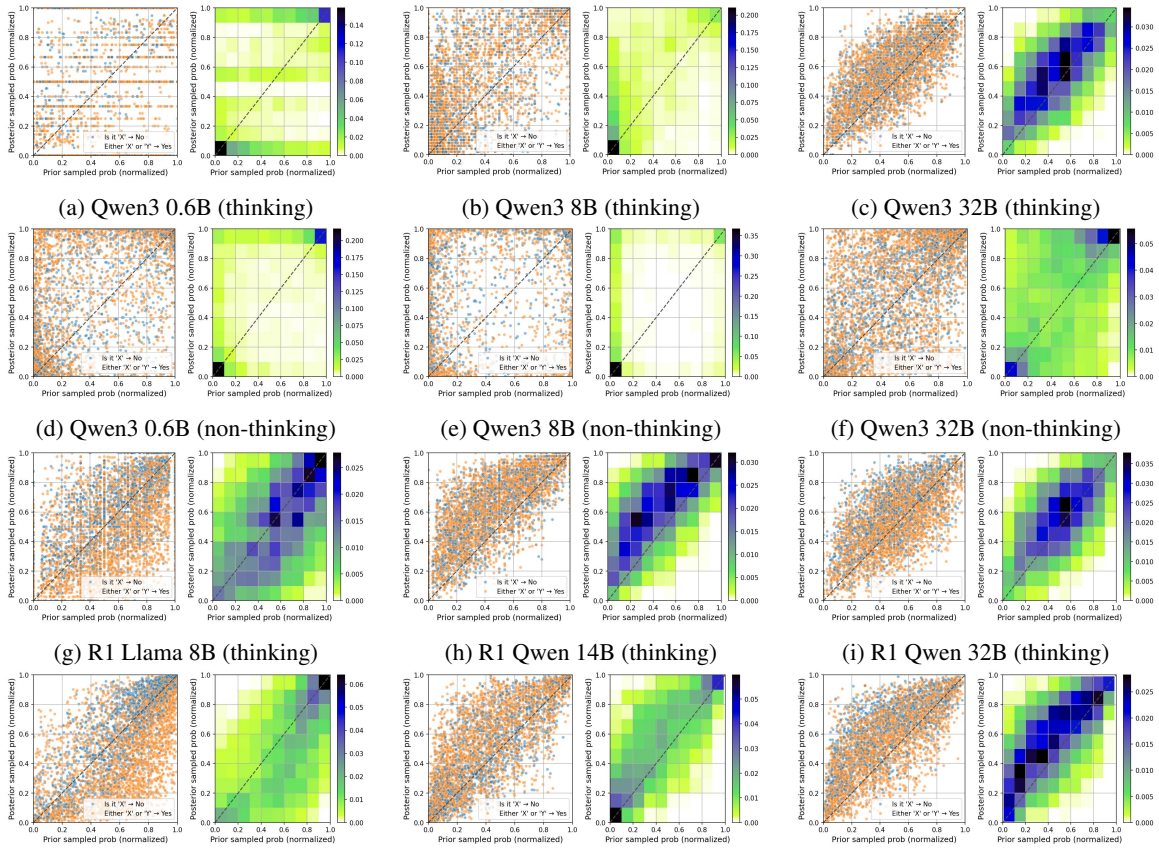
(g) Qwen2.5-Coder 0.5B (h) Qwen2.5-Coder 7B (i) Qwen2.5-Coder 32B

(j) DeepSeek LLM 7B (k) DeepSeek Coder 7B (l) DeepSeekMath 7B

Figure 4: Instance-level 2-class normalized probability plots and heatmaps for non-reasoning models. The horizontal axis represents the normalized probability in the prior scenario for the first not rejected option, $\overline{p_\alpha^{\text{prior}}}$, and the vertical axis represents the normalized probability in the posterior scenario, $\overline{p_\alpha^{\text{posterior}}}$, as defined in Equation 2. If consistency holds, the points align along the $y = x$ line. Neither the Llama 3.1 family nor the Qwen2.5 family shows clear improvement with scaling. The Qwen2.5-Coder family exhibits increased entropy and convergence toward the $y = x$ line compared to the corresponding Qwen2.5 models. DeepSeek Coder and DeepSeekMath demonstrate more consistent behavior than the base DeepSeek LLM model.

set answers. We calculated normalized probabilities for the closed-set answers from the corresponding log-probabilities of the option tokens. The results are shown in the Table 8. As the table shows, the effects of continual pre-training on coding and mathematics corpora and model size scaling on calibration are inconsistent. While the calibration score improves with scaling in Llama models, in the Qwen2.5 family, the calibration score is generally better for smaller models. For coding training, Qwen2.5 shows improvement in all models except the 32B model, which worsened, while DeepSeek models show no clear trend. As mentioned in the previous section, continual pre-training on coding and mathematics corpora tends to strongly affect the entropy regularization of the free-form generation, but in some cases, it has little effect on the closed-set side, which is thought to lead to this lack of consistency in the results.

The calibration plots are shown in Figures 6 and 7. As a note, no rejection-pattern–specific plot is shown for the calibration at the prior context, as no entity rejection occurs at this stage. In calibration, the influence of prompt differences at the posterior context was not strongly observed; models with good calibration are well-calibrated with either prompt, and bad models are bad with either prompt. The calibration scores from continual pre-training on coding and mathematics corpora show no consistent variation, but they tend to have a stronger influence on the entropy regularization of the open-set output, while in contrast, the closed-set side is often unaffected. Scaling has a positive effect in Llama 3.1, but no consistent effect can be seen in Qwen2.5.



(a) Qwen3 0.6B (thinking) (b) Qwen3 8B (thinking) (c) Qwen3 32B (thinking)

(d) Qwen3 0.6B (non-thinking) (e) Qwen3 8B (non-thinking) (f) Qwen3 32B (non-thinking)

(g) R1 Llama 8B (thinking) (h) R1 Qwen 14B (thinking) (i) R1 Qwen 32B (thinking)

(j) R1 Llama 8B (non-thinking) (k) R1 Qwen 14B (non-thinking) (l) R1 Qwen 32B (non-thinking)

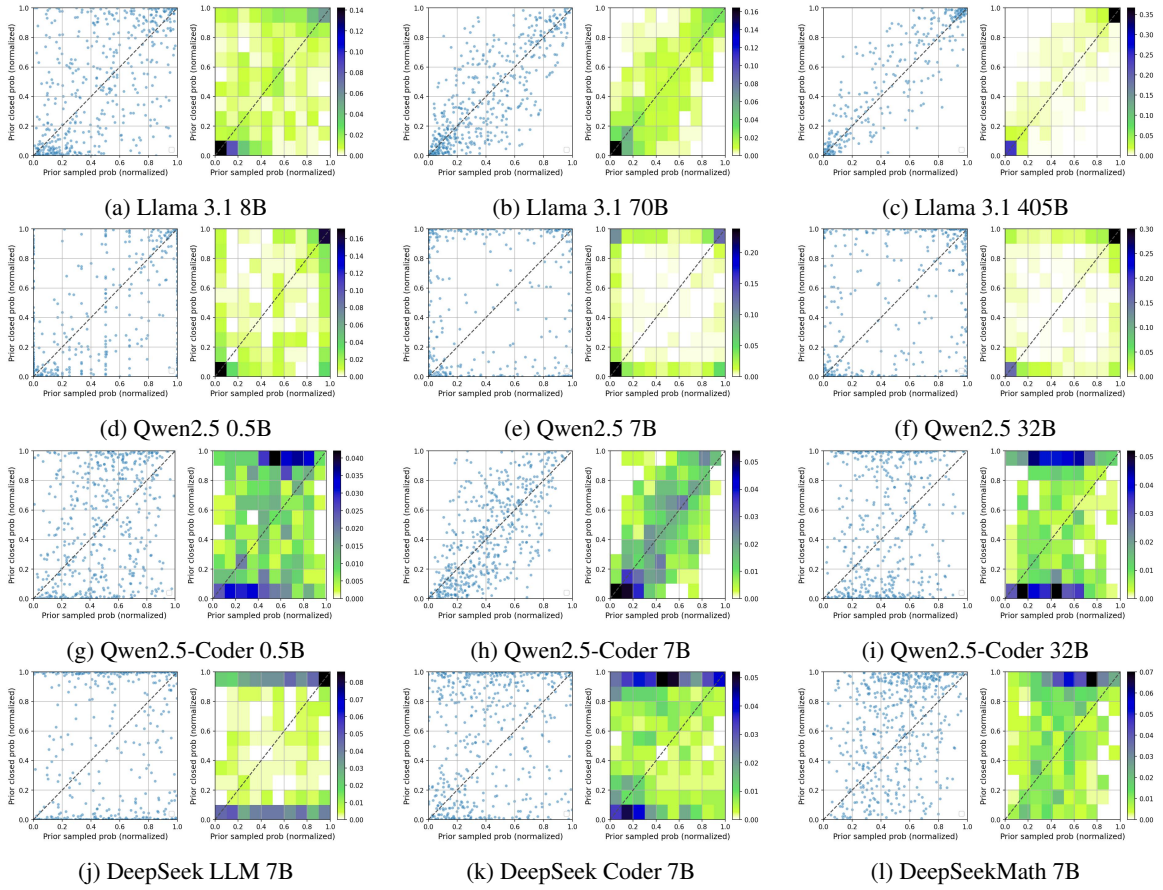
Figure 5: Instance-level 2-class normalized probability plots and heatmaps for reasoning models. Each figure is presented in the same manner as in Fig. 4. Qwen3 models exhibit distinct behaviors depending on whether the thinking mode is enabled or disabled. In contrast, DeepSeek R1 models show more moderate differences.

E Error Analysis

The Qwen2.5 and Qwen2.5-Coder 0.5B models generate a large number of verbalization errors at the prior context. This is largely due to errors where the model fails as a LLM by not understanding the prompt. DeepSeek Coder 7B also has many verbalization errors at the prior context, but these errors are thought to be strongly rooted in instruction tuning preferences rather than LLM errors, with refusals like 'as I am AI assistant'. As a general trend, verbalization error is lower at the posterior than at the prior, with significant improvement even in models that had high errors at the prior. This is likely because the context of one turn of conversation establishes a flow of following instructions. In Qwen2.5 0.5B, a peak concentrating at a specific probability occurs at the prior context. This is because many verbalization errors occurred at the prior context, and the low entropy resulted in cases where some options were almost never counted even with 5000 samples, leading to many instances having the same selection ratio.

E.1 Analysis of Cases Where an Inappropriate Option is Chosen

Here, we provide a detailed analysis of cases where an inappropriate option is chosen in the posterior. We investigated the correlation coefficients and weighted probabilities of choosing an inappropriate option, based on the hypothesis that if an option that becomes inappropriate in the posterior was already frequently chosen in the prior, it is more likely to be chosen as an inappropriate option in the posterior. Table 9 shows the detailed analysis results for cases where an inappropriate option is chosen. Although no choice is inappropriate at the prior stage, here we use $p_{\text{invalid}}^{\text{prior}}$ for convenience to denote the selection probability in the prior for the option that corresponds to the inappropriate option in the posterior for that instance. In the table, "Norm." is the value when normalized as a 3-class probability distribution excluding verbal errors. "Weighted Norm." is the value obtained by taking a weighted average of the normalized prior and posterior probabilities, using the normal-



(a) Llama 3.1 8B (b) Llama 3.1 70B (c) Llama 3.1 405B

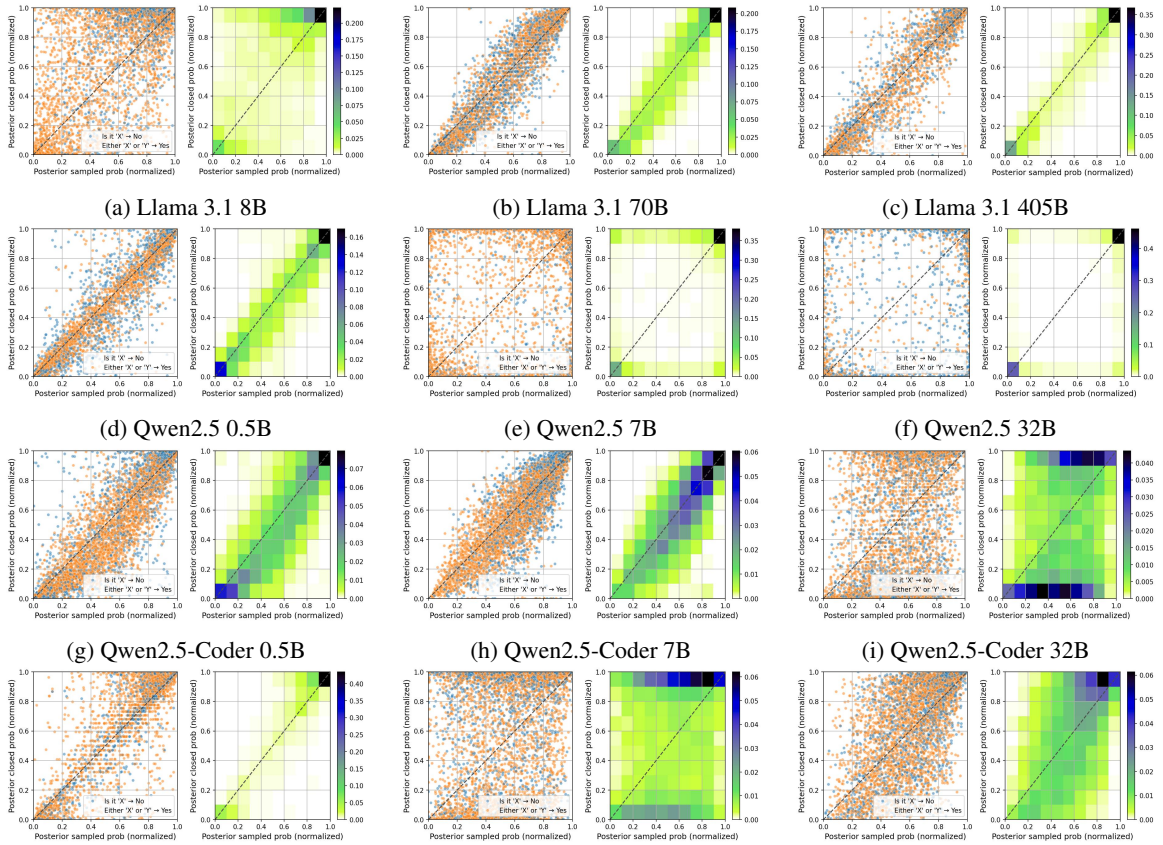
(d) Qwen2.5 0.5B (e) Qwen2.5 7B (f) Qwen2.5 32B

(g) Qwen2.5-Coder 0.5B (h) Qwen2.5-Coder 7B (i) Qwen2.5-Coder 32B

(j) DeepSeek LLM 7B (k) DeepSeek Coder 7B (l) DeepSeekMath 7B

Figure 6: Calibration plots and heatmaps at the prior context.

ized probability of the other as the weight. It is displayed to evaluate how much the target value varies when the value used for weighting is large. "Increase Rate" is the value after weighted normalization divided by its own normalized weight. Due to symmetry, the value is the same whether the prior or posterior is used as the base. A value greater than 1 indicates that in cases where an invalid option is frequently chosen in the prior/posterior, the probability of choosing that option in the other turn is also high. In this experiment only, we did not exclude instances where no appropriate options were sampled in the prior/posterior; we only excluded cases where none of the three options were sampled, and then measured the probabilities. Therefore, in most instances, the normalized prior probability is 33.33% due to the symmetry of the posterior patterns. However, as an exception, symmetry does not hold when cases occur where no option is sampled in some posteriors, so for some models, the value is not 33.33%. Correlation coefficients were measured between the 3-class normalized values of the prior and posterior. Since Llama 3.1 405B never chooses an inappropriate option, some values are omitted. Qwen2.5 32B is almost 0 but sometimes chose an inappropriate option (0.000004%). Although the Pearson correlation coefficient is negative for this model, it is unreliable because in most cases, no inappropriate options are chosen. Excluding these, the correlation coefficient is positive in all cases, confirming a tendency that when an inappropriate option is sampled in the posterior, the corresponding option was also sampled at a relatively high probability in the prior, supporting the hypothesis. However, the level varies considerably by model, and it also differs from the trend of the increase rate. Therefore, it cannot necessarily be said that the posterior is strongly influenced only by the simple magnitude of the selection probability at the prior stage. For example, in Qwen2.5-Coder 7B, although the correlation coefficient is high, the increase rate is not that large. It can be said that the magnitude relationship is maintained, but even in cases where many invalid options are chosen in the posterior, the probability of choosing that option in the prior is not outstandingly high.



(a) Llama 3.1 8B (b) Llama 3.1 70B (c) Llama 3.1 405B

(d) Qwen2.5 0.5B (e) Qwen2.5 7B (f) Qwen2.5 32B

(g) Qwen2.5-Coder 0.5B (h) Qwen2.5-Coder 7B (i) Qwen2.5-Coder 32B

(j) DeepSeek LLM 7B (k) DeepSeek Coder 7B (l) DeepSeekMath 7B

Figure 7: Calibration plots and heatmaps at the posterior context.

F Other Statistics

Table 10 shows the number of contexts used to calculate the score for each model. Smaller models struggled to generate appropriate responses, and in some cases, the appropriate options were never sampled and therefore were excluded. For larger models, certain options in some contexts were never sampled in the prior scenario due to low entropy, making evaluation of those contexts impossible.

Model	Think	Calibration (2-class)	
		Prior	Posterior
Llama 3.1 8B		0.9310	0.9216
Llama 3.1 70B		0.9624	0.9858
Llama 3.1 405B		**0.9836**	0.9879
Qwen2.5 0.5B		0.8600	0.9838
Qwen2.5 1.5B		0.9214	0.9427
Qwen2.5 3B		0.7992	0.7468
Qwen2.5 7B		0.7737	0.8738
Qwen2.5 14B		0.6755	0.8743
Qwen2.5 32B		0.8521	0.9243
Qwen2.5-Coder 0.5B		0.8826	0.9699
Qwen2.5-Coder 1.5B		0.9764	0.9776
Qwen2.5-Coder 3B		0.9764	0.9846
Qwen2.5-Coder 7B		0.9552	0.9782
Qwen2.5-Coder 14B		0.9546	0.9613
Qwen2.5-Coder 32B		0.8292	0.8591
DeepSeek LLM 7B		0.7502	0.9749
DeepSeek Coder 7B		0.8261	0.8418
DeepSeekMath 7B		0.9130	0.9424
Qwen3 0.6B	✓	0.8869	0.7770
Qwen3 1.7B	✓	0.6973	0.7975
Qwen3 4B	✓	0.6945	0.7277
Qwen3 8B	✓	0.7906	0.7813
Qwen3 14B	✓	0.8198	0.8380
Qwen3 32B	✓	0.8845	0.8901
Qwen3 0.6B		0.9434	0.9517
Qwen3 1.7B		0.9832	0.9832
Qwen3 4B		0.8296	0.7821
Qwen3 8B		0.7708	0.8917
Qwen3 14B		0.8555	0.9445
Qwen3 32B		0.9210	0.9590
DeepSeek R1 Llama 8B	✓	0.8361	0.8404
DeepSeek R1 Qwen 14B	✓	0.8904	0.8720
DeepSeek R1 Qwen 32B	✓	0.9130	0.9522
DeepSeek R1 Llama 8B		0.8521	0.8735
DeepSeek R1 Qwen 14B		0.9431	0.8830
DeepSeek R1 Qwen 32B		0.9715	**0.9882**

Table 8: Calibration performance evaluated by the proposed consistency score.

Model	Think	$p_{\text{invalid}}^{\text{prior}}$			$p_{\text{invalid}}^{\text{posterior}}$			Increase Rate	$p_{\text{invalid}}^{\text{prior}}$ vs $p_{\text{invalid}}^{\text{posterior}}$	
		Raw	Norm.	Weighted Norm.	Raw	Norm.	Weighted Norm.		Pearson	Spearman
Llama 3.1 8B		32.90%	33.33%	62.45%	2.50%	2.73%	5.11%	187.36%	0.3719	0.4899
Llama 3.1 70B		32.83%	33.33%	53.11%	2.82%	2.85%	4.53%	159.34%	0.2726	0.2649
Llama 3.1 405B		33.13%	33.33%	-	**0.00%**	**0.00%**	0.00%	-	-	-
Qwen2.5 0.5B		0.52%	33.33%	42.29%	14.55%	21.94%	27.83%	126.86%	0.2310	0.2329
Qwen2.5 1.5B		30.12%	33.33%	52.49%	27.37%	27.62%	43.49%	157.46%	0.6619	0.6887
Qwen2.5 3B		32.66%	33.33%	35.00%	32.27%	32.66%	34.29%	105.00%	0.0342	0.1641
Qwen2.5 7B		33.29%	33.33%	**75.55%**	1.45%	1.46%	3.32%	**226.66%**	0.2289	0.4722
Qwen2.5 14B		32.69%	33.25%	74.36%	0.04%	0.07%	0.15%	223.64%	0.0287	0.0748
Qwen2.5 32B		33.33%	33.33%	24.92%	0.00%	0.00%	0.00%	74.76%	-0.0033	0.0041
Qwen2.5-Coder 0.5B		1.32%	33.33%	38.85%	6.59%	28.14%	32.80%	116.56%	0.2080	0.3631
Qwen2.5-Coder 1.5B		21.85%	33.33%	39.36%	27.86%	29.71%	35.08%	118.08%	0.2841	0.3788
Qwen2.5-Coder 3B		30.08%	33.33%	37.58%	18.61%	19.62%	22.12%	112.74%	0.1741	0.2667
Qwen2.5-Coder 7B		32.58%	33.33%	48.07%	12.97%	13.31%	19.19%	144.21%	**0.6855**	**0.7232**
Qwen2.5-Coder 14B		25.68%	33.33%	43.87%	13.02%	13.99%	18.41%	131.61%	0.5230	0.5012
Qwen2.5-Coder 32B		33.16%	33.33%	42.27%	12.05%	12.08%	15.32%	126.81%	0.3646	0.3625
DeepSeek LLM 7B		15.94%	33.33%	66.61%	19.15%	22.67%	45.31%	199.83%	0.7625	0.7566
DeepSeek Coder 7B		7.51%	33.33%	45.48%	9.66%	11.67%	15.93%	136.43%	0.3220	0.3846
DeepSeekMath 7B		28.96%	33.33%	43.21%	17.59%	21.88%	28.37%	129.63%	0.3975	0.4721
Qwen3 0.6B	✓	23.59%	32.66%	44.32%	4.73%	26.01%	35.30%	135.71%	0.2838	0.2553
Qwen3 1.7B	✓	30.98%	33.33%	46.17%	5.82%	9.22%	12.77%	138.52%	0.2701	0.2674
Qwen3 4B	✓	32.54%	33.33%	46.70%	0.76%	0.79%	1.10%	140.10%	0.1233	0.1849
Qwen3 8B	✓	32.66%	33.33%	49.62%	0.76%	0.77%	1.15%	148.87%	0.1068	0.1426
Qwen3 14B	✓	33.07%	33.33%	59.04%	0.27%	0.27%	0.48%	177.12%	0.1456	0.2127
Qwen3 32B	✓	32.38%	33.33%	43.98%	2.23%	2.29%	3.02%	131.95%	0.3231	0.2968
Qwen3 0.6B		30.98%	33.31%	50.30%	11.95%	27.58%	41.65%	151.01%	0.3651	0.3798
Qwen3 1.7B		33.24%	33.33%	78.04%	25.50%	25.51%	59.73%	234.12%	0.5838	0.6487
Qwen3 4B		32.95%	33.32%	49.94%	11.74%	11.88%	17.81%	149.89%	0.1733	0.2338
Qwen3 8B		32.94%	33.33%	66.26%	11.45%	11.53%	22.91%	198.77%	0.3900	0.5629
Qwen3 14B		33.33%	33.33%	67.47%	1.66%	1.69%	3.42%	202.41%	0.2077	0.3536
Qwen3 32B		22.31%	33.33%	52.22%	2.20%	2.26%	3.54%	156.67%	0.2711	0.4606
DeepSeek R1 Llama 8B	✓	15.18%	33.33%	39.81%	17.00%	22.57%	26.95%	119.42%	0.3226	0.3619
DeepSeek R1 Qwen 14B	✓	22.29%	33.33%	45.58%	7.56%	8.37%	11.45%	136.73%	0.5600	0.5526
DeepSeek R1 Qwen 32B	✓	28.83%	33.33%	41.61%	6.69%	7.37%	9.20%	124.83%	0.4746	0.4346
DeepSeek R1 Llama 8B		11.07%	33.33%	42.15%	22.05%	27.89%	35.26%	126.44%	0.3639	0.4338
DeepSeek R1 Qwen 14B		27.22%	33.33%	47.74%	24.81%	26.65%	38.17%	143.23%	0.6568	0.6831
DeepSeek R1 Qwen 32B		22.19%	33.33%	46.64%	16.00%	16.79%	23.49%	139.93%	0.6571	0.6723

Table 9: Weighted invalid probabilities. The weight is the invalid probabilities at the posterior context, and the value is the corresponding option probabilities at the prior context. Note that some values for Llama 3.1 405B are omitted since its $p_{\text{invalid}}^{\text{posterior}}$ is exactly 0.

Model	Think	Entity Likelihood Rank-N	
		1-1000	10001-11000
maximum		4500	4500
Llama 3.1 8B		4500	4500
Llama 3.1 70B		4500	4500
Llama 3.1 405B		4494	4500
Qwen2.5 0.5B		3972	4023
Qwen2.5 1.5B		4500	4500
Qwen2.5 3B		3474	3384
Qwen2.5 7B		4494	4488
Qwen2.5 14B		4392	4473
Qwen2.5 32B		4470	4488
Qwen2.5-Coder 0.5B		4500	4500
Qwen2.5-Coder 1.5B		4500	4500
Qwen2.5-Coder 3B		4500	4500
Qwen2.5-Coder 7B		4500	4500
Qwen2.5-Coder 14B		4500	4500
Qwen2.5-Coder 32B		4500	4500
DeepSeek LLM 7B		4500	4500
DeepSeek Coder 7B		4494	4500
DeepSeekMath 7B		4500	4500
Qwen3 0.6B	✓	3741	
Qwen3 1.7B	✓	4477	
Qwen3 4B	✓	4470	
Qwen3 8B	✓	4470	
Qwen3 14B	✓	4485	
Qwen3 32B	✓	4500	
Qwen3 0.6B		4453	
Qwen3 1.7B		3189	
Qwen3 4B		4278	
Qwen3 8B		4400	
Qwen3 14B		4491	
Qwen3 32B		4500	
DeepSeek R1 Llama 8B	✓	4500	
DeepSeek R1 Qwen 14B	✓	4500	
DeepSeek R1 Qwen 32B	✓	4500	
DeepSeek R1 Llama 8B		4500	
DeepSeek R1 Qwen 14B		4500	
DeepSeek R1 Qwen 32B		4500	

Table 10: Numbers of evaluated posteriors. For each posterior, 500 samples were taken, and since each setting has 9 posterior variations, the maximum number in each row is 4500.