Program-Aided Reasoners (*Better*) Know What They Know

Anonymous ACL submission

Abstract

Prior work shows that program-aided reasoning, in which large language models (LLMs) are combined with programs written in programming languages such as Python, can significantly improve accuracy on various reasoning tasks. However, while accuracy is essential, it is also important for such reasoners to "know what they know", which can be quantified through the *calibration* of the model. In this paper, we compare the calibration of Program Aided Language Models (PAL) and text-based Chain-of-thought (COT) prompting techniques over 5 datasets and 2 model types - LLaMA models and OpenAI models. Our results indicate that PAL leads to improved calibration in 75% of the instances. Our analysis uncovers that prompting styles that produce lesser diversity in generations also have more calibrated results, and thus we also experiment with inducing lower generation diversity using temperature scaling and find that for certain temperatures, PAL is not only more accurate but is also more calibrated than COT. Overall, we demonstrate that, in the majority of cases, program-aided reasoners *better* know what they know than text-based counterparts.[1]

1 Introduction

As language models (LMs) grow in size and capabilities, several works examine methods to improving their reasoning skills with different styles of prompting (Wei et al., 2022; Wang et al., 2022; Suzgun et al., 2022b; Zhou et al., 2022; Yao et al., 2023). One representative method, chain of thought (COT) reasoning (Wei et al., 2022), takes inspiration from how humans approach problem-solving – by breaking down the problem into a sequence of natural language explanations before arriving at a final answer. Furthermore, prompts that enable problem-solving are not limited to natural language; program-aided language models (PAL); Gao et al. (2022) have demonstrated the efficacy of using code (such as Python programs) as a means of improving the model's reasoning, surpassing the accuracy of conventional chain-of-thought style prompts in some tasks (Madaan et al., 2022; Lyu et al., 2023; Zhang et al., 2023a,b). Both methods are illustrated in Figure 1.

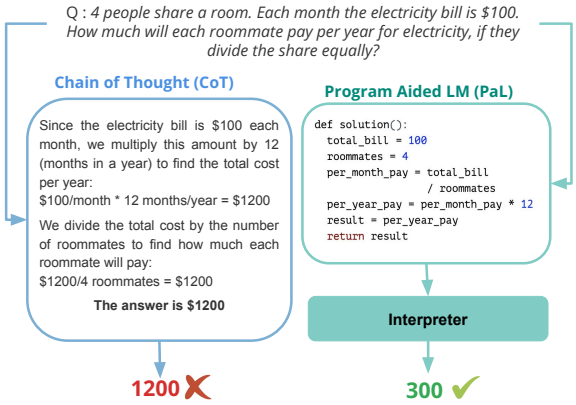


Figure 1: Comparisons of COT and PAL outputs. COT can sometimes generate the correct reasoning chain but fail to derive the correct answer as a final step. PAL fixes this issue by executing generated code to arrive at a deterministic answer.

Currently, most works proposing such methods have been primarily focused on improving accuracy. However, for real-world applications, another highly desirable feature of ML systems is that they should be able to provide *reliable confidence estimates*. Accurate estimates of model confidence are helpful for many applications, including allowing the model to refrain from providing an answer when uncertain, asking for human intervention in uncertain cases, or providing confidence estimates to a downstream model that consumes the outputs. The reliability is measured through *calibration*, how a model's confidence in its predictions aligns accurately with actual outcomes (Guo et al., 2017a; Jiang et al., 2020; Zhao et al., 2021).

[1]Anonymized code and data are available at https://anonymous.4open.science/r/code-calibrates-A61D/.

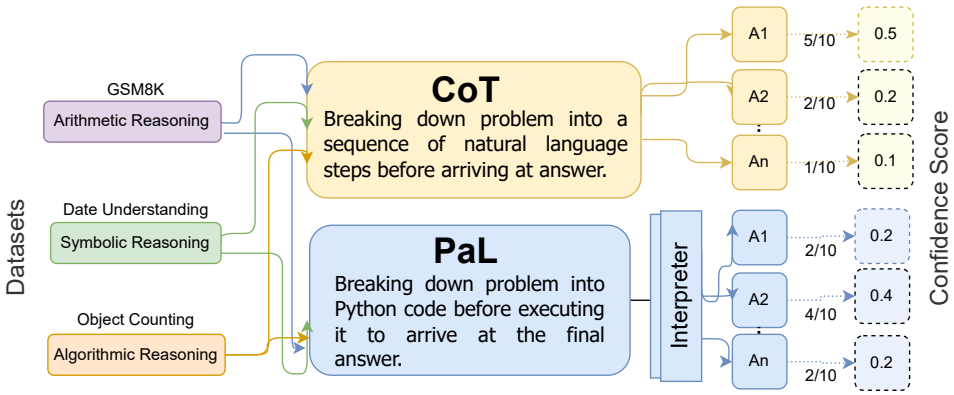


Figure 2: Illustration of eliciting model confidence through self-consistency

In sum, the previous research has shown, as eloquently stated by Kadavath et al. (2022) "language models *(mostly)* know what they know" — LLMs are reasonably well calibrated, although some imperfections remain.

In this work, we examine the effect of program-aided reasoning on calibration. We consider five datasets that cover different reasoning tasks and evaluate the performance of both PAL and COT style prompting for OpenAI models (OpenAI, 2023) and LLaMA models (Touvron et al., 2023) with respect to accuracy and calibration. We primarily explore three main research questions :

- ***RQ 1:*** *Does program-aided reasoning result in significantly different calibration than text-based* COT*?*
- ***RQ 2:*** *Are the observed trends different across OpenAI models and LLaMA models?*
- ***RQ 3:*** *Does the consistency of LLM generations affect calibration? We examine this by measuring generation diversity and answer space entropy.*

Our results show that program-aided reasoners know what they know *even better* than standard text-based reasoners with COT. In particular, on OpenAI models, PAL exhibits not only superior accuracy but also a consistent enhancement in calibration of about 50%, over COT. Interestingly, the consistent improvement of calibration is not observed in LLaMA models. Still, we find that adjusting the temperature of sampling (similar to a widely used method of Platt scaling (Platt et al., 1999), PAL improves with respect to accuracy and calibration. We also conduct a detailed analysis of these observations and find a correlation between the similarity of the generated chains-of-thoughts or programs and calibration, which might help explain these trends. Code and data available here under the Apache 2.0 license.

2 Preliminaries and Mathematical Formulation

2.1 Measuring Calibration

Calibration refers to the alignment between the predicted probability estimates of a model and their actual correctness or accuracy (Guo et al., 2017b). Formally, a perfectly calibrated model can be expressed using the following equation, where X is the given input, Y is the true output, the model's output is $\hat{Y}$ and $P_N(\hat{Y} \mid X) = p$ is the probability, or "confidence", over the model's output.

$$P\left(\hat{Y} = Y \mid P_N(\hat{Y} \mid X) = p\right) = p, \forall p \in [0, 1] \quad (1)$$

In essence, Equation 1 conveys that if a perfectly calibrated model makes 100 predictions, and the confidence of each prediction is 0.6, then we expect the accuracy to be also 0.6. Nevertheless, the model may exhibit varying confidence levels for each sample. Therefore, it is imperative to calculate calibration across all confidence scores. We estimate this probability by dividing the predictions into M separate and equally sized interval buckets based on their confidence levels.

We use the expected calibration error (ECE), a common measure of (lack of) calibration, a weighted average of the discrepancy between each bucket's accuracy and confidence. It is given in Equation 2

Here B_m is the m-th bucket that contains samples whose probabilities of predictions fall in the

interval $\left(\frac{m-1}{M}, \frac{m}{M}\right]$, where $\frac{|B_m|}{n}$ is B_m's size relative to all the samples. $\operatorname{acc}(B_m)$ is the average accuracy of the samples in the *m*-th bucket, and $\operatorname{conf}(B_m)$ is the corresponding average confidence of the samples falling in the *m*-th bucket.

$$\sum_{m=1}^{M} \frac{|B_m|}{n} \left|\operatorname{acc}(B_m) - \operatorname{conf}(B_m)\right| \quad (2)$$

Consider a setup where we have buckets with a width of 0.1. All instances where a model assigns probabilities between 0.4 and 0.5 will be allocated to the bucket B_5 or the bucket encompassing probabilities between 0.4 and 0.5. Subsequently, the average accuracy for the instances in these buckets and the average probability/confidence is computed. The absolute difference of the accuracy and confidence is multiplied by the proportion of total instances in a bucket. This process is repeated for every bucket, and the individual scores are summed up to calculate ECE.

2.2 Self-consistency as a measure of confidence

Self-consistency (Wang et al., 2022) is a natural language reasoning technique that uses chain-of-thought prompting to generate multiple paths for reasoning. This process aims to select the most consistent answer by sampling and marginalizing. Here, we use a latent variable Z to represent the reasoning chain/programs. Y is the answer that is either extracted in case of CoT or obtained after execution in case of PaL. We marginalize over Z by taking a majority vote over answers. Thus, we rely on majority voting over the answers to obtain confidence estimates for each sample.

K is a hyperparameter that controls the number of generations (referenced in equation 3). The higher the value of K, the better our approximation of the probability of each sample. Figure 2 shows an overview of this process.

$$P(\hat{Y}_0|Z_0) = \frac{1}{K}\sum_{i=0}^{K} \mathbb{I}\left\{\hat{Y}_i = \hat{Y}_0\right\} \quad (3)$$

Wang et al. (2022) and Xiong et al. (2023a) suggest that self-consistency can be an effective way to elicit confidence from models. Hence, given the lack of per-token log probabilities in closed LMs like `gpt-3.5-turbo` and `text-davinci-003`, we adopt self-consistency as a proxy measure for calibration.

2.3 Similarity and Answer Entropy

In addition to empirically evaluating the impact on accuracy and calibration, we conduct a qualitative analysis of the reasoning chains (the latent variable Z described previously). Here, we observe a consistent pattern, i.e. the correct answers corresponding to a question are often associated with similar generations.

We find that this observation aligns with the finding made by Li et al. (2022a), that there are numerous ways in which solutions can be incorrect. In contrast, correct solutions tend to exhibit more uniform behaviour.

To empirically validate this observation, we employed sentence embeddings generated from the *all-MiniLM-v6* model to compute the average similarity among the generations/reasoning chains, equivalent to calculating similarity over latent variables Z.

Furthermore, to gain deeper insights into the relationship between similarity in generations and corresponding answers, we compute the entropy $H(A)$ of the answer space where $P(a_i)$ refers to the probability of the ith answer in K answers obtained by extraction or program execution for a given sample.

$$H(A) = -\sum_{i=1}^{K} P(a_i) \cdot \log_2 P(a_i) \quad (4)$$

This allowed us to investigate whether the observed similarity in the latent variable space Z leads to a lower entropy within the answer space.

3 Experimental Design

3.1 Models

We compare the calibration and accuracy of two different prompting strategies - CoT and PaL on an equal number of closed-source and open-source models. The open source models used in experimentation are `LLaMA2-13B`, `LLaMA2-70B` (Touvron et al., 2023). and the closed-source models are `gpt-3.5-turbo`, `text-davinci-003` (Brown et al., 2020). It should be noted that all models have received some form of supervision from code during pre-training (OpenAI, 2023; Touvron et al., 2023), in addition to being primarily trained on text. For LLaMA models, we leveraged vLLM (Kwon et al., 2023) for distributed inference using A6000 GPU(s).

Dataset	Category	# Samples	Example
GSM8K (Cobbe et al., 2021)	Arithmetic	1319	Q: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take? A: 3
GSM8K Hard (Gao et al., 2022)	Arithmetic	1319	Q: A robe takes 2287720 bolts of blue fiber and half that much white fiber. How many bolts in total does it take? A: 3431580
Date Understanding (Suzgun et al., 2022a)	Symbolic	360	Q: Yesterday was April 30, 2021. What is the date today in MM/DD/YYYY? A: 05/01/2021
Object Counting (Suzgun et al., 2022a)	Algorithmic	250	Q: I have three couches, a lamp, a stove, a table, a fridge, and a microwave. How many objects do I have? A: 8
Repeat Copy (Suzgun et al., 2022a)	Algorithmic	32	Q: say python twice and data once, and then repeat all of this three times. A: python python data python python data python python data

Table 1: Datasets with their examples and categories.

3.2 Hyperparameters

For our experiments, we set temperature (T) as 1.0 and the probability (p) for nucleus sampling (Holtzman et al., 2020) as 1.0. Selecting a temperature of 1.0 enables direct sampling from the model as no scaling of probabilities is involved, as seen from Equation 5. Here, z_i refers to the logit for the ith token generated, and N is the vocabulary size.

$$\sigma\left(z_i\right)=\frac{e^{\frac{z_i}{T}}}{\sum_{j=0}^{N} e^{\frac{z_j}{T}}} \tag{5}$$

For use $K = 10$ generations per sample for all datasets. We set the maximum number of tokens (input + output) for each generation to 1024.

3.3 Tasks

We examined reasoning tasks encompassing several challenges, including arithmetic, algorithmic, and symbolic reasoning. We use five datasets that cover these different kinds of reasoning tasks. The arithmetic reasoning datasets include *GSM8K* (Cobbe et al., 2021) and *GSM8K Hard* (Gao et al., 2022). The algorithmic reasoning tasks include *Object-Counting* (Suzgun et al., 2022a) and *Repeat-Copy* (Suzgun et al., 2022a). We used *Date-Understanding* as a Symbolic Reasoning Dataset (Suzgun et al., 2022a). Specific information about the datasets used can be found in Table 1.

3.4 Prompt Design

We provide all models with natural language chain-of-thought (CoT) prompts and code-based Program-Aided Language Model (PaL) prompts. For datasets where CoT prompts are available in their original form, we use them as presented in the original paper (Wei et al., 2022). We modify these prompts for other datasets to suit the specific task while maintaining their original format. For PaL prompts, we use and adapt the code prompts provided in (Gao et al., 2022). The prompts are included in Appendix A.

4 Results

We investigate two model types: OpenAI models and LLaMA models along with the two different prompting strategies - PAL and CoT.

4.1 Effect of prompting style on Calibration

In this section, we look at the first two *RQs*:

RQ 1: *Does one prompting style result in significantly better calibration than the other?*

RQ 2: *Are the observed calibration trends different across OpenAI models and LLaMA models?*

Table 2 shows results for OpenAI models, we observe that PAL prompting improves both calibration and accuracy across all datasets. We see approximately 50% relative reduction in calibration error and an average improvement of 18.42% in accuracy.

In Figure 3, we show reliability plots which illustrate improved calibration, with the reliability curves for PAL prompting consistently aligning closer to the ideal reliability curve as compared to CoT across datasets. While PAL shows a notable gain of 14.83% in accuracy across all datasets for LLaMA models, it shows better calibration in only half of our settings. Overall, for both OpenAI models and LLaMA models, we observe that PAL leads to better calibration than CoT for 75% of the settings. The reliability plots for all datasets for the models `gpt-3.5-turbo` and `LLaMA2-70B` can be seen in Appendix Section E, D.

Effect of PAL on calibration controlling for accuracy One reasonable hypothesis is that PAL is improving calibration because it achieves higher accuracy, and more accurate models can be better calibrated. To examine this hypothesis, we conduct statistical analysis using *mixed linear models* (McLean et al., 1991), which allows us to consider

Name	Score	Model	GSM8K		Object-Counting		Repeat-Copy		Date-Understanding		GSM8K Hard	
			CoT	PaL	CoT	PaL	CoT	PaL	CoT	PaL	CoT	PaL
LLaMA2-70B	ECE (↓)	LLaMA	0.19	0.07	0.17	0.14	0.18	0.23	0.09	0.18	0.07	0.03
	ACC (↑)	LLaMA	59.28	63.91	76.00	92.40	40.62	71.88	66.66	70.18	21.45	40.62
	SIM (↑)	LLaMA	72.20	92.40	94.43	94.72	87.10	90.58	86.87	82.15	92.28	74.32
	ENT (↓)	LLaMA	2.24	1.92	1.00	0.76	1.93	2.00	1.44	1.54	2.85	2.17
LLaMA2-13B	ECE (↓)	LLaMA	0.06	0.08	0.08	0.06	0.11	0.17	0.06	**0.05**	0.12	0.14
	ACC (↑)	LLaMA	27.0	34.34	56.4	81.6	34.37	53.12	48.24	50.41	6.67	25.55
	SIM (↑)	LLaMA	76.6	93.3	93.2	95.3	89.8	88.6	79.5	84.2	74.0	92.32
	ENT (↓)	LLaMA	2.83	2.49	1.52	0.85	2.43	2.47	2.23	2.06	2.42	3.06
text-davinci-003	ECE (↓)	OpenAI	0.04	0.03	0.29	**0.02**	0.20	**0.06**	0.19	0.11	0.15	0.07
	ACC (↑)	OpenAI	65.65	76.49	59.21	**98.00**	67.23	**93.75**	60.70	72.35	23.95	**71.27**
	SIM (↑)	OpenAI	90.5	97.8	99.1	99.8	96.2	98.2	92.4	97.4	89.8	97.9
	ENT (↓)	OpenAI	1.27	0.79	0.36	0.02	1.38	0.44	0.71	0.64	2.31	0.81
gpt-3.5-turbo	ECE (↓)	OpenAI	0.05	**0.03**	0.38	0.03	0.18	0.16	0.17	0.13	0.13	**0.05**
	ACC (↑)	OpenAI	**84.00**	82.40	82.40	97.20	56.25	68.75	61.51	**77.23**	55.21	62.91
	SIM (↑)	OpenAI	94.40	97.80	99.10	98.60	97.70	97.90	95.3	97.6	90.60	95.40
	ENT (↓)	OpenAI	0.57	0.49	0.59	0.048	1.15	0.35	0.50	0.36	1.65	2.43

Table 2: Comparison of Expected Calibration Error (ECE (↓)) , Accuracy (ACC (↑)) , Cosine Similarity (SIM (↑)) and Answer Entropy (ENT (↓)) across datasets. The darker blue shade highlights better performing prompting technique.

the significance of varying the prompting strategy while controlling for accuracy as a confounding factor.

Upon analyzing the results in Table 3, we observe that, when treating the prompting style as a fixed effect, PAL exhibits a negative coefficient of -0.103 (p=0.0) for OpenAI models, which is statistically significant with a threshold of p=0.05. This implies that PAL contributes to the reduction in ECE and has a positive impact on calibration. On the contrary, for LLaMA models, we did not find that PAL had a statistically significant effect on ECE after controlling for accuracy. Across LLaMA models and OpenAI models, PAL has a statistically significant (p=0.02) correlation of -0.067 with ECE, indicating that PAL helps increase calibration on the whole even when controlling for accuracy.

To summarize, we see that PAL prompting has better calibration than CoT prompting (***–RQ1***) . While PAL has improved calibration in all settings for OpenAI models, this trend is less consistent for LLaMA models (***–RQ2***).

Model Type	LLaMA models	OpenAI models	Both
Fixed Effect (ECE vs Prompting Style)	PAL : -0.010	PAL : -0.103	PAL : -0.067
p-value	0.961	0.000	0.002

Table 3: Statistical analysis using mixed linear models, keeping ECE vs Prompting Style as a fixed effect and accuracy as a random effect.

4.2 Effect of generation diversity on calibration

In this section, we look at the third research question: ***RQ 3:*** *Does the consistency of LLM generations affect calibration?*

Qualitative analysis of the generations reveals that PAL generations adhere to a consistent structure that divides the problem-solving process into three distinct parts. This is depicted in Figure 4. In the first part, the model initializes the variables and sets up their initial values required for the calculation. This part is straightforward due to syntactic constraints and remains largely similar across generations. In the second part, the model generates the required logic by applying formulas and utilizing various operations to derive the desired result. Finally, in the third part, the model generates the answer by assigning the calculated result to a variable and returning it, which, again, doesn't vary much across generations. Hence, the diversity of the generation is mainly limited to the second part, making code more constrained in its generation space compared to text. Therefore, there is a **standardized structure in the code** generated by language models with PaL prompts.

Lower generation diversity and answer entropy observed in prompting strategy with better calibration To quantitatively analyze if code-based generations have lower generation diversity and lead to a narrower answer space, we computed aggregated cosine similarity scores for all the generations and entropy over the answer space. For OpenAI models, we note that the cosine similarity scores with PAL are higher than the corresponding scores for CoT. This observation suggests that code-based generations display a higher degree of similarity from a semantic perspective. Moreover, the answer entropy for PAL is lower than CoT. This implies that similar generations that cluster together in the semantic space (Li et al., 2022a) also converge to the equivalent solution space. This

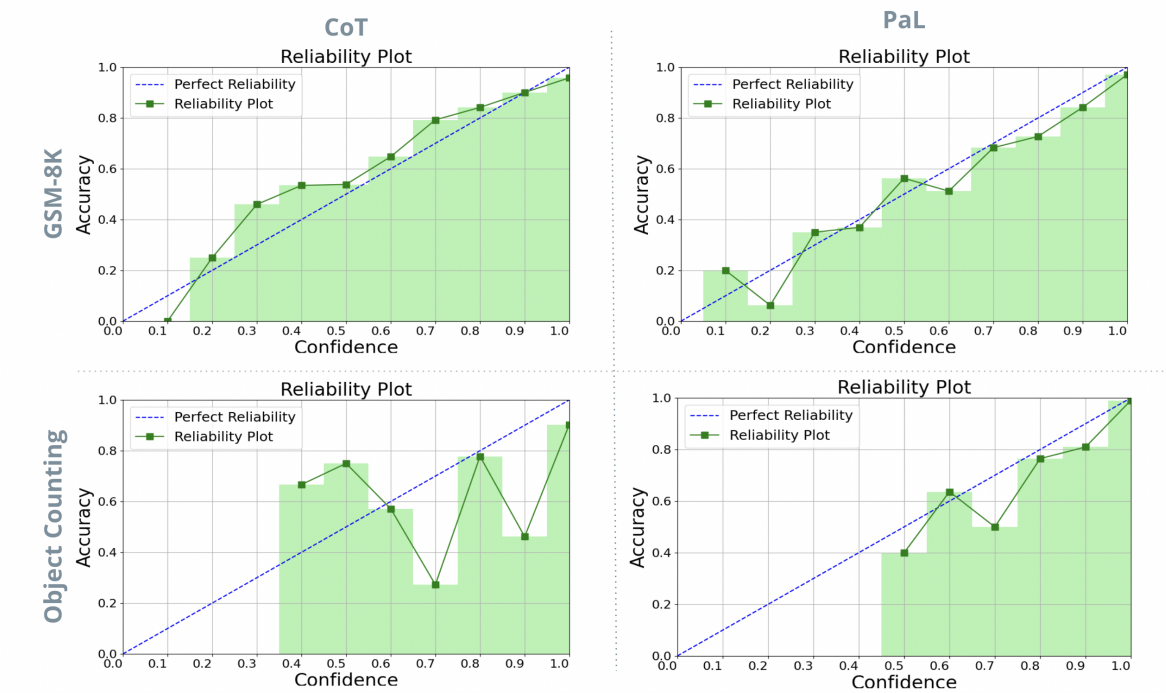


Figure 3: Reliability Plots for various structured reasoning tasks for the model `gpt-3.5-turbo`. The x-axis represents confidence, and the y-axis represents accuracy.

```
def solution ()  :
    # Part 1: Initialize
    num_glasses = 16
    first_glass_price = 5
    second_glass_discount = 0.6

    # Part 2: Calculate
    second_glass_price = first_glass_price *
                         second_glass_discount
    pair_price = first_glass_price +
                 second_glass_price
    num_pairs = num_glasses // 2
    total_cost = num_pairs * pair_price

    # Part 3: Result Generation
    result = total_cost
    return result
```

Figure 4: Typical output structure with PaL

leads to lower uncertainty in the probability distribution of the answer space and, hence, lower entropy. From Table 2, we thus can see that PAL helps produce similar generations that converge to the same answer space, which is also consistently correct. Hence, it achieves better performance and provides more confidence in its predictions.

For LLaMA models, we don't see this trend of PAL having higher generation similarity and lower answer entropy for all datasets. However, for almost all settings for LLaMA models and OpenAI models, the prompting strategy that produces more similar generations and lower answer entropy is also more calibrated.

To summarize, it is evident that lower generation diversity and lower answer entropy are correlated with higher calibration. (***–RQ3***)

Better calibration observed for PAL when inducing similarity in generations for LLaMA models We observe that for OpenAI models, PAL is not only more accurate but also more calibrated than COT. Consequently, we explore whether the reduction in generation diversity, achievable through lower temperatures, can contribute to improved calibration for LLaMA models.

We perform a parameter sweep across temperature values between 0.1 and 0.7 with a step size of 0.2. We show the variation of accuracy, calibration, generation similarity, and answer entropy for two datasets in Figure 5. The plots for the remaining datasets are available in Appendix B, Figure 6. We can see that we obtain better calibration for both the LLaMA models in both PAL and COT for temperatures below 1.0. From Tables 4 and 5, we note that in the majority of runs with T < 1.0, PAL is better calibrated than COT. Considering accuracy and calibration, optimal performance is achieved at different temperatures for each dataset. For most T values, the similarity scores are higher while corresponding answer entropy values are lower for PAL compared to COT. This mirrors the pattern

Temp		GSM8K		Object-Counting		Repeat-Copy		Date-Understanding		GSM8K Hard	
		CoT	PaL	CoT	PaL	CoT	PaL	CoT	PaL	CoT	PaL
0.7	ECE	0.101	0.07	0.076	0.03	0.14	0.12	0.12	0.09	0.18	0.03
	ACC	66.03	67.9	77.6	93.2	53.1	75.0	74.5	76.42	27.14	52.91
	SIM	85.07	97.47	98.53	99.42	93.78	94.81	89.62	96.16	83.28	97.29
	ENT	1.60	1.48	0.55	0.21	1.46	1.35	0.88	0.80	2.43	1.72
0.5	ECE	0.049	0.036	0.103	0.059	0.112	0.075	0.114	0.063	0.139	0.104
	ACC	66.94	67.24	77.23	92.4	59.3	68.75	73.44	77.2	27.7	51.63
	SIM	88.69	98.25	99.17	99.85	97.09	96.81	92.49	97.97	87.65	98.2
	ENT	1.35	1.19	0.39	0.12	1.09	0.99	0.60	0.52	2.18	1.39
0.3	ECE	0.057	0.097	0.140	0.064	0.194	0.113	0.153	0.139	0.230	0.206
	ACC	64.89	63.38	78.8	91.2	53.12	71.87	72.62	76.42	26.16	49.28
	SIM	91.91	98.75	99.51	99.94	97.73	98.27	95.18	99.02	91.14	98.75
	ENT	1.087	0.960	0.238	0.056	0.780	0.504	0.420	0.317	1.866	1.076
0.1	ECE	0.219	0.257	0.188	0.07	0.278	0.156	0.233	0.176	0.418	0.380
	ACC	58.6	58.37	77.2	90.4	53.12	68.75	69.91	78.32	23.5	45.87
	SIM	95.79	99.37	99.82	99.98	99.28	99.64	98.21	99.68	95.31	99.35
	ENT	0.661	0.526	0.085	0.026	0.288	0.173	0.195	0.137	1.179	0.540

Table 4: Results of temperature scaling for `LLaMA2-70B`. The darker blue shade highlights better performing prompting technique.

observed for OpenAI models. For `LLaMA-2 13b`, PAL displays better calibration than CoT at lower temperatures. However, the optimal temperature for obtaining the best performance for calibration and accuracy is still T=1.0.

For `LLaMA-2 70b`, optimal temperature values in our runs for calibration are either 0.5 or 0.7, while extreme values (0.1, 1.0) yield lower calibration and accuracy performance. We can, therefore see that scaling temperatures in the LLaMA models can help us to obtain better calibration for PAL, specifically for the `LLaMA-2 70b`, which already performs better than CoT on these reasoning tasks. Thus, we do see that lower generation diversity and lower answer entropy lead to higher calibration up to a certain point, after which it negatively affects the calibration. (***–RQ3***)

5 Related Work

5.1 Prompting Strategies for Reasoning

Recent developments in language models have introduced various methods to enhance their reasoning abilities. One such method is CoT (Wei et al., 2022), which helps models generate a series of intermediate steps to solve problems. CoT has demonstrated improved performance in arithmetic, common sense, and symbolic reasoning tasks. There are approaches such as PaL (Gao et al., 2022) and Program-of-thoughts (PoT) (Chen et al., 2022), which go a step further by generating programs as intermediate steps and using an interpreter to process them. Code as a medium of reasoning has shown considerable promise, evidenced by better performance over chain-of-thought style prompting strategies in several recent studies (Madaan et al., 2022; Gao et al., 2022; Lyu et al., 2023; Zhang et al., 2023a,b). Unlike these works, our primary goal in this paper is to understand the effect of code prompts on calibration.

5.2 Calibration in Language Models

Calibration has been extensively studied in structured prediction problems, such as named entity recognition and part of speech tagging (Jagannatha and Yu, 2020), as well as in natural language understanding tasks, like question answering and text classification (Kamath et al., 2020; Kong et al., 2020; Desai and Durrett, 2020). More recently, studies have focused on calibrating language models when used as generators (Jiang et al., 2021; Zhao et al., 2021). Additionally, the study by Kadavath et al. (2022) explored the likelihood of a model knowing the answer before proposing a response. However, these approaches typically rely on access to the model's logits.

In contrast, the work by (Tian et al., 2023) investigates verbalized probability estimates to assess the calibration of large language models without needing access to logits. This involves querying the model about its confidence in the answers it generates. Furthermore, (Xiong et al., 2023b) introduced self-consistency-based methods for calibration, demonstrating their superior performance compared to verbalized methods. In our research, we adopt self-consistency as the method of choice for measuring calibration.

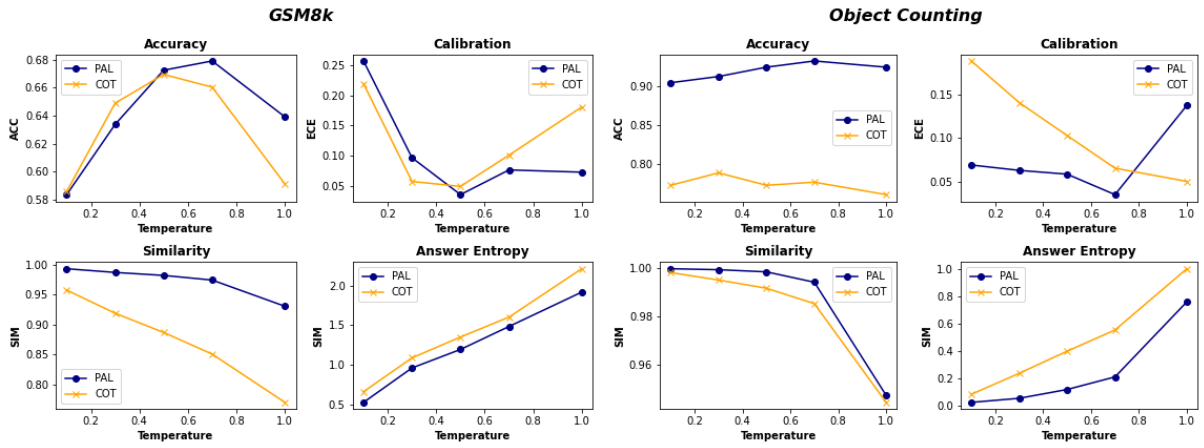


Figure 5: Trends seen in temperature scaling for the model `LLaMA2-70B`. Across datasets, the accuracy and calibration improve upon lowering the temperature to a certain extent. This is in line with having lower generation similarity and lower answer entropy. The optimal temperatures seen are 0.5 and 0.7 across datasets. For other datasets, refer Appendix, Figure 6.

5.3 Utilizing Language for Code Generation

The exploration of using natural language for code generation has taken diverse approaches in research. Initial efforts involved rule-based, predictive and deep-learning variations (Gulwani and Marron, 2014; Woods, 1973; Zelle and Mooney, 1996; Lin et al., 2017; Rabinovich et al., 2017). However, performance enhancements were observed using pre-trained models trained on code-based datasets (Chen et al., 2021; Nijkamp et al., 2022; Gao et al., 2022; Li et al., 2023). Employing pre-trained language models (LMs) for code generation as a way to solve tasks that require step-by-step structuring and various forms of reasoning has proven to be particularly effective (Ni et al., 2023a; Gao et al., 2022; Ni et al., 2023b).

Intermediate execution results from code have been used for training (Chen et al., 2018) and inference (Wang et al., 2018). Majority-based voting on the results of code executions (which is the self-consistency-based methodology we employ) has also been shown to be an effective technique for selecting the right candidate (Li et al., 2022b; Cobbe et al., 2021; Shi et al., 2022).

6 Conclusion

In this study, we explore the impact of two distinct prompting styles, namely PAL and CoT, on the calibration of OpenAI models and LLaMA models. Our investigation spans 5 reasoning datasets, employing self-consistency as the methodology for eliciting calibration. We analyze four different metrics - calibration (ECE) , accuracy (ACC) , average similarity in generations (SIM) , and answer entropy (ENT) . Our findings are as follows:

- ***RQ 1:*** *Does one prompting style result in significantly better calibration than the other?* Empirical results show that PAL generally has higher calibration and accuracy for 82.5% of the cases across OpenAI and LLaMA models for a varied range of temperatures.

- ***RQ 2:*** *Are the observed calibration trends different across OpenAI models and LLaMA models?* We observed that OpenAI models are in general better calibrated for the reasoning tasks with up to 19% improvement in ECE.

- ***RQ 3:*** *Does the consistency of LLM generations affect performance?* PAL prompting shows a general trend of having greater similarity in the generation over CoT, which we hypothesize could be due to the inherent structure present in the code. We see that greater generation similarity is accompanied by lower answer entropy and lower ECE.

We hope that this study will catalyze additional research aimed at holistically evaluating and gaining deeper insights into the role of prompts in various tasks and domains.

7 Limitations

Access to OpenAI models is only available through an API which limits the ability to exactly control the hyperparameters influencing the genera-

tions. Moreover, OpenAI models are not transparent, which limits the ability to study these models. Because of this lack of transparency it is also hard to draw conclusive insights about any comparisons between OpenAI models and LLaMA models In our study, we report the results from a single run but due to combination of utilizing temperature value of 1.0 and hardware induced stochasticity, it is possible to get varying results for a given model.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *ArXiv*, abs/2211.12588.

Xinyun Chen, Chang Liu, and Dawn Xiaodong Song. 2018. Execution-guided neural program synthesis. In *International Conference on Learning Representations*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. *arXiv preprint arXiv:2003.07892*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models. *ArXiv*, abs/2211.10435.

Sumit Gulwani and Mark Marron. 2014. Nlyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 803–814.

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

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017b. On calibration of modern neural networks. In *International Conference on Machine Learning*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration.

Abhyuday Jagannatha and Hong Yu. 2020. Calibrating structured output predictors for natural language processing. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2020, page 2078. NIH Public Access.

Zhengbao Jiang, J. Araki, Haibo Ding, and Graham Neubig. 2020. 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.

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.

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

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. *arXiv preprint arXiv:2006.09462*.

Lingkai Kong, Haoming Jiang, Yuchen Zhuang, Jie Lyu, Tuo Zhao, and Chao Zhang. 2020. Calibrated language model fine-tuning for in-and out-of-distribution data. *arXiv preprint arXiv:2010.11506*.

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

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022a. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Yujia Li, David H. Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom, Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de,

Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey, Cherepanov, James Molloy, Daniel Jaymin Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de, Freitas, Koray Kavukcuoglu, and Oriol Vinyals. 2022b. Competition-level code generation with alphacode. *Science*, 378:1092 – 1097.

Xi Victoria Lin, Chenglong Wang, Deric Pang, Kevin Vu, and Michael D Ernst. 2017. Program synthesis from natural language using recurrent neural networks. *University of Washington Department of Computer Science and Engineering, Seattle, WA, USA, Tech. Rep. UW-CSE-17-03-01*.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*.

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. *ArXiv*, abs/2210.07128.

Robert A McLean, William L Sanders, and Walter W Stroup. 1991. A unified approach to mixed linear models. *The American Statistician*, 45(1):54–64.

Ansong Ni, Srini Iyer, Dragomir Radev, Veselin Stoyanov, Wen-tau Yih, Sida Wang, and Xi Victoria Lin. 2023a. Lever: Learning to verify language-to-code generation with execution. In *International Conference on Machine Learning*, pages 26106–26128. PMLR.

Ansong Ni, Pengcheng Yin, Yilun Zhao, Martin Riddell, Troy Feng, Rui Shen, Stephen Yin, Ye Liu, Semih Yavuz, Caiming Xiong, Shafiq R. Joty, Yingbo Zhou, Dragomir R. Radev, and Arman Cohan. 2023b. L2ceval: Evaluating language-to-code generation capabilities of large language models. *ArXiv*, abs/2309.17446.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. A conversational paradigm for program synthesis.

OpenAI. 2023. Openai documentation. `https://platform.openai.com/docs/model-index-for-researchers`.

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. *arXiv preprint arXiv:1704.07535*.

Freda Shi, Daniel Fried, Marjan Ghazvininejad, Luke Zettlemoyer, and Sida I. Wang. 2022. Natural language to code translation with execution. *ArXiv*, abs/2204.11454.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022a. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Mirac Suzgun, Nathan Scales, Nathanael Scharli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Huai hsin Chi, Denny Zhou, and Jason Wei. 2022b. Challenging big-bench tasks and whether chain-of-thought can solve them. *ArXiv*, abs/2210.09261.

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*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. 2018. Robust text-to-sql generation with execution-guided decoding. *arXiv: Computation and Language*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

William A Woods. 1973. Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, pages 441–450.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2023a. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *ArXiv*, abs/2306.13063.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2023b. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Li Zhang, Liam Dugan, Hai Xu, and Chris Callison-Burch. 2023a. Exploring the curious case of code prompts. *ArXiv*, abs/2304.13250.

Li Zhang, Hai Xu, Yue Yang, Shuyan Zhou, Weiqiu You, Manni Arora, and Chris Callison-Burch. 2023b. Causal reasoning of entities and events in procedural texts. In *Findings*.

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.

Denny Zhou, Nathanael Scharli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Huai hsin Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *ArXiv*, abs/2205.10625.

A Prompts

The following sections display one example of the few-shot prompts used for each dataset across prompting styles.

A.1 PAL Prompts

A.1.1 GSM8K/GSM8K-Hard

```
def solution ()  :
    """Olivia has $23. She bought five bagels for $3 each. How much money does she have left?"""
    money_initial = 23
    bagels = 5
    bagel_cost = 3
    money_spent = bagels * bagel_cost
    money_left = money_initial - money_spent
    result = money_left
    return result
```

A.1.2 Object Counting

```
# Q: I have a chair, two potatoes, a cauliflower, a lettuce head, two tables, a cabbage, two
↪  onions, and three fridges. How many vegetables do I have?
```
```
def solution ()  :
        # note: I'm not counting the chair, tables, or fridges
        vegetables_to_count = {{'potato': 2,'cauliflower': 1,'lettuce head': 1,'cabbage':
        ↪  1,'onion': 2}}
        return sum (vegetables_to_count.values () )
```

A.1.3 Date Understanding

```
# Q: 2015 is coming in 36 hours. What is the date one week from today in MM/DD/YYYY?
# If 2015 is coming in 36 hours, then today is 36 hours before.
today = datetime (2015, 1, 1)  - relativedelta (hours=36)
# One week from today,
one_week_from_today = today + relativedelta (weeks=1)
# The answer formatted with %m/%d/%Y is
one_week_from_today.strftime ('%m/%d/%Y')
```

A.1.4 Repeat Copy

```
# Q: Repeat the word duck four times, but halfway through also say quack
```
```
def solution ()  :
        result = []
        for i in range (1, 5) :
                result.append ("duck")
                if i == 2:
                        result.append ("quack")
        return " ".join (result)
```

A.2 CoT Prompts

A.2.1 GSM8K/GSM8K-Hard

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6. The answer is 6.

A.2.2 Object Counting

Q: I have a chair, two potatoes, a cauliflower, a lettuce head, two tables, a cabbage, two onions, and three fridges. How many vegetables do I have?
A: Chair, tables and fridges are not vegetables, so we are not counting them. Two potatoes, cauliflower, lettuce head, cabbage and two onions are vegetables, so they will be counted The total number of vegetables are 2 + 1 + 1 + 1 + 2 The answer is: 7

A.2.3 Date Understanding

Q: 2015 is coming in 36 hours. What is the date one week from today in MM/DD/YYYY?
A: If 2015 is coming in 36 hours, then it is coming in 2 days. 2 days before 01/01/2015 is 12/30/2014, so today is 12/30/2014. So one week from today will be 01/05/2015. So the answer is 01/05/2015.

A.2.4 Repeat Copy

Q: Repeat the word duck four times, but halfway through also say quack
A: On repeating the word duck four times we get: duck duck duck duck Halfway through if we say quack, we have to say quack in between the 2nd word and the 3rd word The answer is: duck duck quack duck duck

B Temperature Scaling Experiments - Line Plots

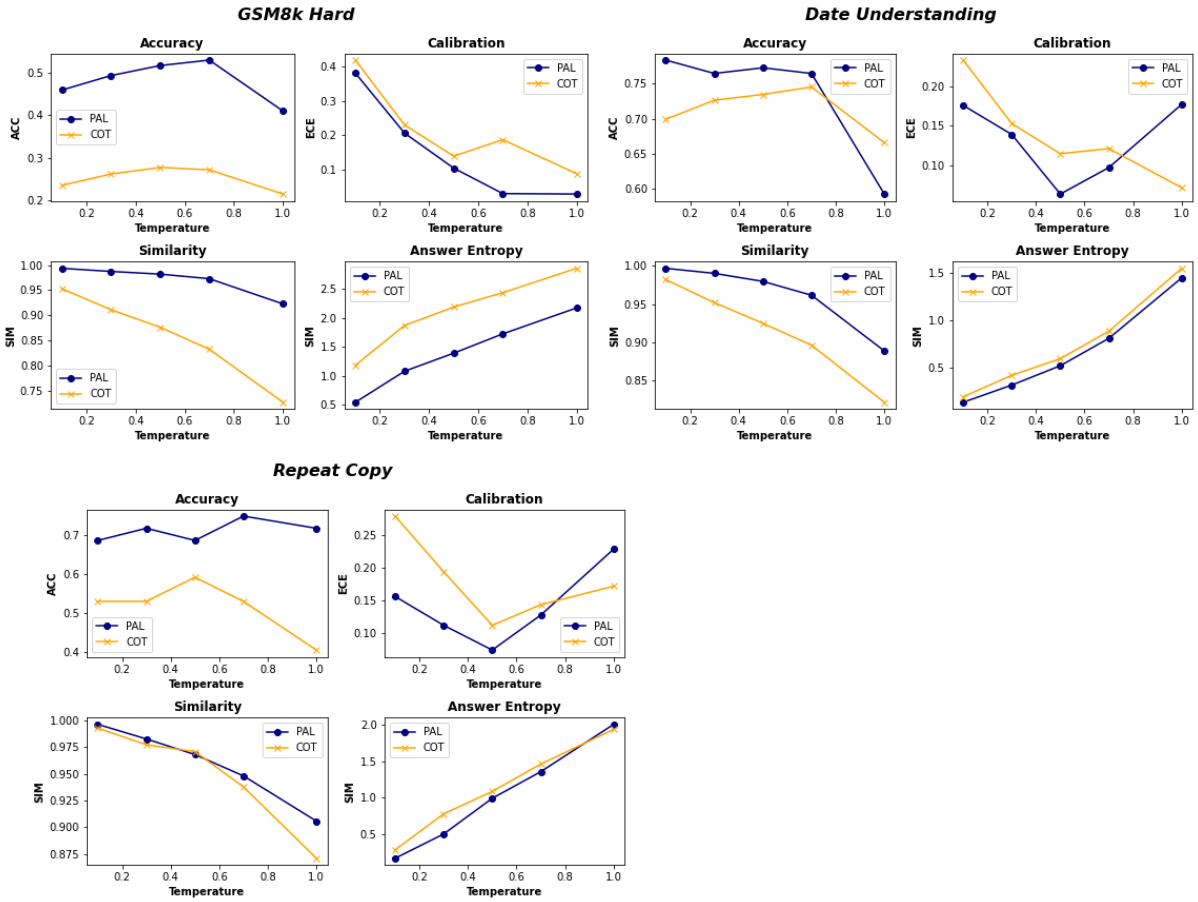


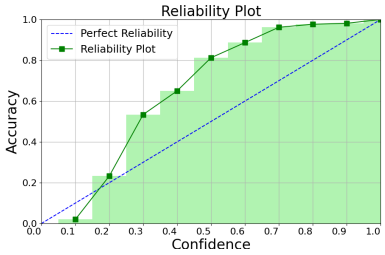
Figure 6: Trends seen in temperature scaling for the datasets - *GSM8K-Hard*, *Date-Understanding* and *Repeat-Copy*

C Results of temperature scaling for LLaMA2-13B

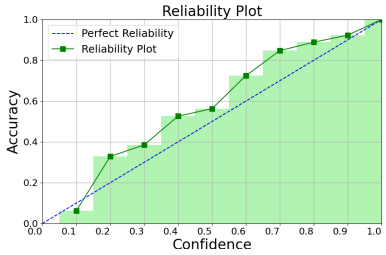
Temp		GSM8K		Object-Counting		Repeat-Copy		Date-Understanding		GSM8K Hard	
		CoT	PaL	CoT	PaL	CoT	PaL	CoT	PaL	CoT	PaL
0.7	ECE	0.052	0.046	0.12	0.108	0.175	0.181	0.108	0.107	0.125	0.087
	ACC	36.16	39.27	61.60	80.80	34.37	59.37	50.60	56.63	10.91	30.32
	SIM	85.40	97.48	98.65	99.33	95.07	94.02	86.81	96.03	83.48	97.37
	ENT	2.405	2.222	0.970	0.303	2.048	1.949	1.505	1.330	2.892	2.332
0.5	ECE	0.099	0.093	0.171	0.135	0.131	0.106	0.134	0.183	0.177	0.1381
	ACC	36.08	37.07	60.80	82.00	34.37	59.30	53.92	55.00	10.80	29.87
	SIM	88.70	98.10	99.21	99.83	97.12	97.74	89.94	97.63	87.37	98.03
	ENT	2.144	1.976	0.722	0.158	1.455	1.311	1.141	0.942	2.694	2.074
0.3	ECE	0.160	0.108	0.231	0.154	0.200	0.206	0.219	0.304	0.246	0.231
	ACC	33.81	30.72	62.40	81.20	37.50	65.63	55.28	50.40	10.16	27.97
	SIM	91.51	98.54	99.56	99.93	97.62	96.81	93.02	98.41	90.48	98.44
	ENT	1.826	1.618	0.475	0.080	0.967	1.19	79.04	63.78	2.389	1.716
0.1	ECE	0.372	0.334	0.311	0.174	0.4969	0.1812	0.341	0.423	0.372	0.334
	ACC	30.25	32.14	62.80	81.20	37.50	46.80	54.47	49.32	30.25	32.14
	SIM	95.12	99.19	99.80	99.98	97.24	98.68	97.00	99.32	95.12	99.19
	ENT	1.145	0.84	0.204	0.014	0.286	0.283	0.373	0.261	1.1458	0.840

Table 5: Results of temperature scaling for LLaMA2-13B. The darker blue shade highlights better performing prompting technique.

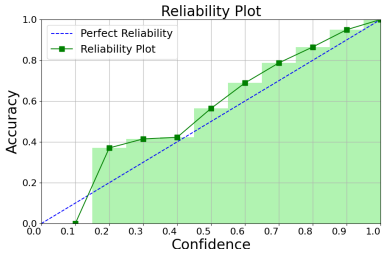
D Reliability Plots for LLaMA2-70B



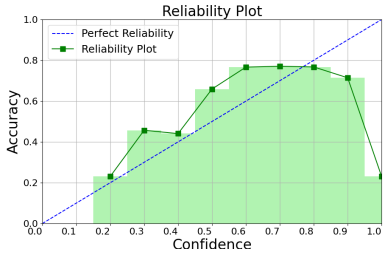
(a) GSM8k CoT



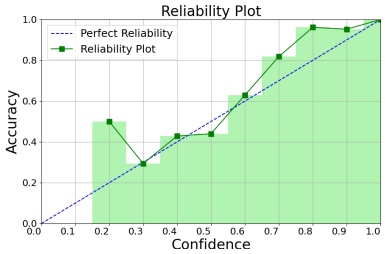
(b) GSM8k PaL



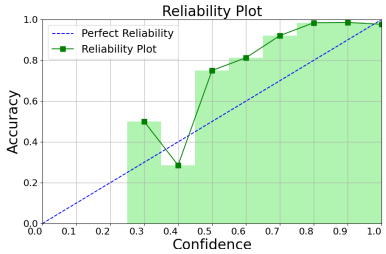
(c) Date Understanding CoT



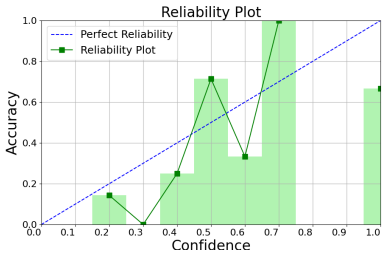
(d) Date Understanding PaL



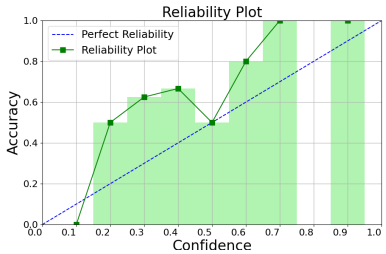
(e) Object Counting CoT



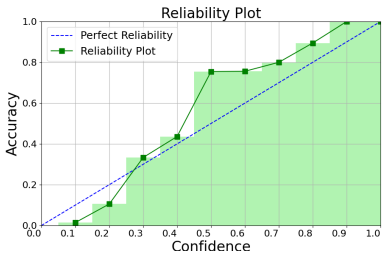
(f) Object Counting PaL



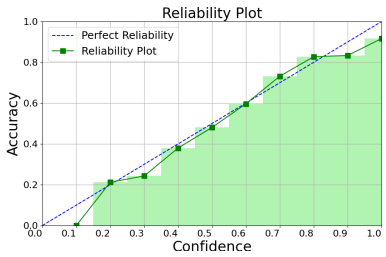
(g) Repeat Copy CoT



(h) Repeat Copy PaL



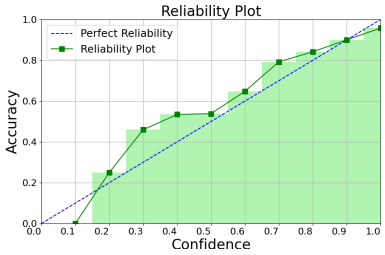
(i) GSM8k Hard CoT



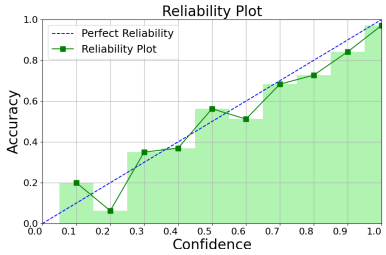
(j) GSM8k Hard PaL

Figure 7: Reliability plots for all the datasets using CoT and PaL prompting for the model LLaMA2-70B

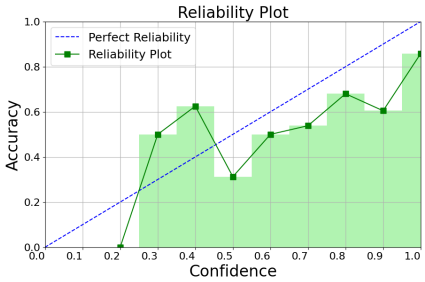
E Reliability Plots for `gpt-3.5-turbo`



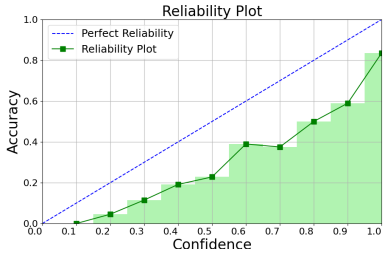
(a) GSM8k CoT



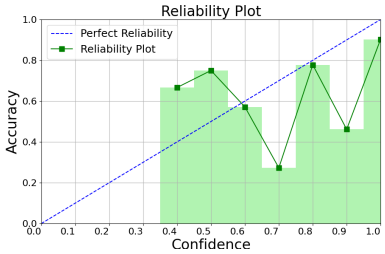
(b) GSM8k PaL



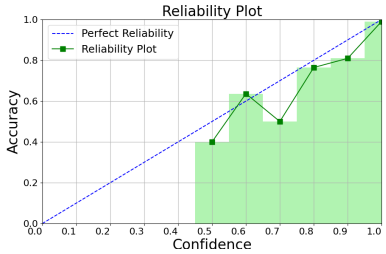
(c) Date Understanding CoT



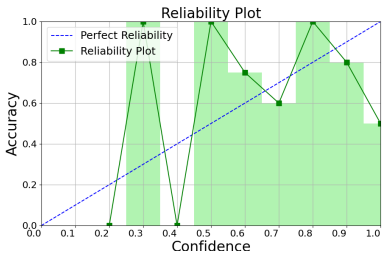
(d) Date Understanding PaL



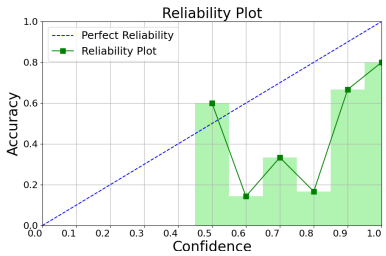
(e) Object Counting CoT



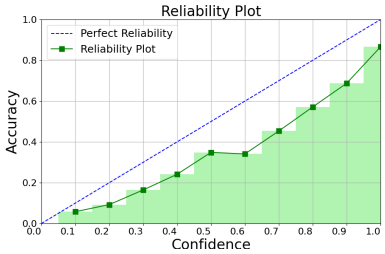
(f) Object Counting PaL



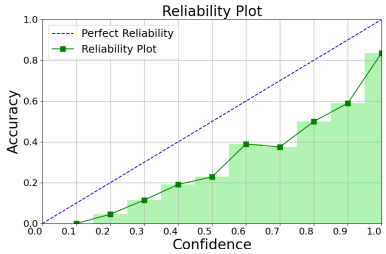
(g) Repeat Copy CoT



(h) Repeat Copy PaL



(i) GSM8k Hard CoT



(j) GSM8k Hard PaL

Figure 8: Reliability plots for all the datasets using CoT and PaL prompting for the model `gpt-3.5-turbo`