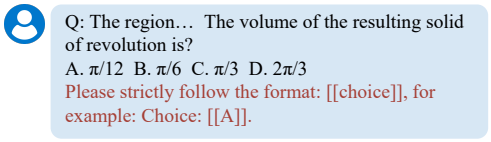


Figure 8: The instruction prompt used to format the LLM output on multiple-choice questions.

searched budget and in the bottom area of Figure 2. We further define such an area as the ideal budget range and give the formalized definition in Section A.1. A good budget should be located in the ideal budget range. Two metrics are taken into consideration: *in-range accuracy* and *out-of-range distance*. In-range accuracy determines whether the predicted budget $\hat{\beta}$ falls within the ideal budget range W_k^*. Mathematically, it can be expressed as:

$$\mathbb{I}\{\hat{\beta} \in W_k^*\} = \begin{cases} 1, & \text{if } \hat{\beta} \in W_k^*, \\ 0, & \text{otherwise.} \end{cases}$$

Out-of-range distance quantifies the distance between $\hat{\beta}$ and W_k^* if the predicted budget β^* falls outside the ideal budget range W_k^*. Let $dist(\hat{\beta}, W_k^*)$ represent the distance, defined as:

$$\text{dist}(\hat{\beta}, W_k^*) = \begin{cases} 0, & \text{if } \hat{\beta} \in W_k^*, \\ \min\limits_{\hat{\beta} \in W_k^*} |\hat{\beta} - \beta|, & \text{if } \hat{\beta} \notin W_k^*. \end{cases}$$

Intuitively, a higher in-range accuracy and a lower out-range distance indicate a better estimated budget. During our evaluation, the in-range accuracy is 60.61%, and the out-of-range distance is 109.64. It indicates that more than two-thirds of estimated budgets are located in the ideal range. For those out-of-range samples, they have an offset of 109.64 tokens on average. Figure 9 illustrates the successful and failed estimated cases intuitively. The prompt we use for budget estimation is as follows:

"`Task: Analyze the given question and estimate the minimum number of tokens required for reasoning.`"

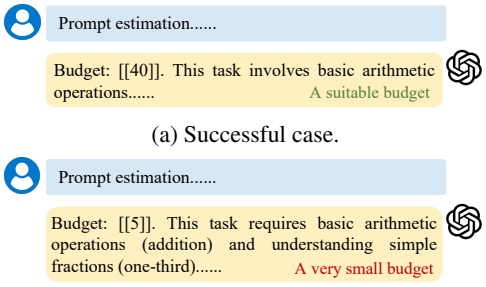
This prompt encourages the model to evaluate the question as a whole, including reasoning depth, structure, completeness, and surface-level difficulty.

A.3 Details of TALE's Implementation

In this section, we introduce the hyper-parameters used for TALE-EP and TALE-PT.

TALE-EP. TALE-EP uses a zero-shot mechanism to estimate the token budget and then prompts the LLM. The instruction prompts used during this process are shown in Figure 7. To ensure output consistency, we set the temperature to 0.1 and limit the model to a single reasoning path. Additionally, the random seed is fixed at 1024.

TALE-PT. TALE-PT includes two implementations: SFT and DPO. For parameter efficiency, both implementations adopt LoRA (Hu et al., 2021) for post-training, with rank set to 8 and lora alpha set to 32. For TALE-PT-SFT, we train for 3 epochs with a batch size of 16, a learning rate of 1e-4, and a weight decay of 0.01. For TALE-PT-DPO, we train for 2 epochs with a batch size of 16, a learning rate of 3e-5, and a weight decay of 0.001.



(a) Successful case.

(b) Failed case.

Figure 9: An intuitive example for successful and failed cases of prompt budget estimation in TALE-EP.

Table 6: Comparison of TALE-EP and TALE-PT.

Metrics	TALE-EP	TALE-PT	
		SFT	DPO
ACC	71.82	78.57	74.11
Output Tokens	112.21	139.63	149.93

A.4 Comparison of TALE-EP and TALE-PT.

In this section, we compare the performance of TALE-EP and TALE-PT (including SFT and DPO). Specifically, we utilize Llama-3.1-8B-Instruct as both the budget estimator for TALE-EP and the base model for TALE-PT, evaluated on GSM8K. Table 6 illustrates the evidence. As the zero-shot-based estimator tends to predict relatively low budgets, TALE-EP achieves lower token usage but at the cost of slightly reduced accuracy. In contrast, both variants of TALE-PT achieve higher accuracy with more tokens, as their training data is constructed using optimal budget search, which enforces answer correctness as a strict constraint even with a higher token costs. This highlights a trade-off between strict correctness preservation (in TALE-PT) and token efficiency (in TALE-EP).

A.5 Applicability of TALE on More Tasks.

To further evaluate the applicability, we deploy

Table 7: Generalization of TALE on more tasks. Three popular LLM generative tasks, Code Summarization (Husain et al., 2019)(CS), Empathetic Response Generation (Rashkin et al., 2019)(ERG), Code Generation (Austin et al., 2021)(CG), are taken into consideration. BLEU is taken as the metric to evaluate the performance. BLEU↑. Output Tokens↑.

Tasks	TALE-EP		Vanilla CoT	
	BLEU	Output Tokens	BLEU	Output Tokens
CS	0.07	44.39	0.2	134.05
ERG	0.005	60.34	0.006	175.37
CG	0.24	171.08	0.267	461.77

TALE-EP on three additional open-ended generative tasks. As these tasks involve open-ended text generation, we adopt the BLEU metric (Papineni et al., 2002) to quantify the similarity between generated outputs and reference texts. The results in Table 4 demonstrate that TALE-EP achieves comparable or even better BLEU scores than Vanilla CoT while using only around 40% of the output tokens, validating its effectiveness and applicability in broader generative tasks.

A.6 Formalizing the Budget Search.

For a given input $\boldsymbol{x}$, we define the search space as:

$$\mathcal{B} = \{\beta \in \mathbb{Z}^+ \mid 0 < \beta \leq T_{\text{vanilla}}(x)\}$$

where $T_{\text{vanilla}}(x)$ is the number of tokens generated by Vanilla CoT. A candidate budget $\beta \in \mathcal{B}$ is considered feasible if:

$$LLM(x, \beta) = y \text{ and } T(x, \beta) < T(x, \beta_0)$$

where y is the ground-truth answer, $T(x, \beta)$ is the actual number of output tokens when answering x under budget β, β_0 is the previously searched (larger) feasible budget. Our goal is to find:

$$\beta^* = \arg\min_{\beta \in \mathcal{B}} T(x, \beta), \text{ subject to } LLM(x, \beta) = y$$

To efficiently find β^*, we employ a binary search procedure guided by the feasibility function above, as detailed in Algorithm 1 and Algorithm 2.

A.7 Efficiency of TALE-EP.

Since TALE-EP requires one additional query, we further evaluate its end-to-end latency in this section. Specifically, we query the Llama-3.1-8B-Instruct model over the GSM8K-Zero dataset and measure both accuracy and average time cost. T he budget estimation query of TALE-EP is also considered. Although TALE-EP requires one additional query compared to Vanilla CoT, it is significantly more efficient, taking only 2.3 seconds per sample, while Vanilla CoT takes 10.2 seconds. This is because the primary factor influencing inference time is the number of output tokens, which TALE-EP effectively reduces.

Table 8: The empirical evidence for "implicit monotonicity assumption". $\bar{\beta}$ is the budget upper bound, which is the token cost of vanilla CoT. The budget row displays scaled budgets ranging from 2^{-2} to $2^2 \cdot \beta^*$.

Budget($*\beta^*$)	2^{-5}	2^{-4}	2^{-2}	2^0
ACC	69.23	75.82	75.82	76.92
Output Tokens	222.69	222.42	244.61	653.53

A.8 Effectiveness of Larger Token Budget.

In scenarios with ample computational resources, the token budget could be larger for better performance. We simulate such a scenario by scaling the estimated budget by a factor α ($\alpha *$ budget). As shown in the table below, increasing α from 1 to 2 leads to higher accuracy (from 67.33% to 72.66%) at the cost of more tokens (from 210.97 to 279.78), demonstrating that TALE-EP can flexibly adapt to different resource scenarios.

A.9 Empirical Evidence for the "Implicit Monotonicity Assumption".

In this section, we give empirical evidence to support the "implicit monotonicity assumption" for our search algorithm. Specifically, for a set of questions, we vary the budget across a range of values (e.g., $2^{-5}, 2^{-4}, 2^{-2}, 2^0$ times budget upper bound, which is the token cost of vanilla CoT), and record the corresponding accuracy and average output tokens at each point. Table 8 illustrates the empirical evidence. Observe that it roughly follows from the monotonicity property. The results demonstrate a consistent trend: accuracy generally increases or plateaus with increasing budgets, confirming a soft monotonicity in most cases.