



Figure 1: Analysis of the impact of CoT on Falcon-40b efficiency: (top) Relation between response time and output length, without CoT (blue dots) and with CoT (red dots), across 100 questions from the GSM8K test set. (bottom) Output words variation between output length with CoT and without CoT using 50 random samples from the GSM8K test set.

applied to decoder only, where the model runs a new inference for each produced word.

To highlight such a dependency, we conducted preliminary tests on Falcon-40B to evaluate the impact of the CoT method in answering arithmetic questions, using a subset of 100 random questions from the GSM8K dataset (Cobbe et al., 2021). The results of this test are illustrated in Figure 1a, where red and blue dots refer to answers given with and without CoT, respectively. The scatter plot shows that CoT significantly increases the output length and generation time. This suggests that while CoT improves the correctness of responses (see Section 7), more attention should be given to the time cost it introduces. To better appreciate the impact of CoT on the output length, Figure 1b reports the output length (in terms of number of generated words) produced by Falcon-40b on a set of 50 questions from GSM8K without CoT (blue bars) and with CoT (pink bars). Note that purple areas denote the areas where the two bars overlap.

#### 4 Metrics for Correct Conciseness

Motivated by the previous considerations, this section presents three novel metrics to evaluate the

capability of an LLM to provide *correct* as well as *concise* responses. The idea is to redefine the classic accuracy metric to integrate conciseness aspects, which as highlighted in Section 7.3 impacts significantly the generation time and the computational cost, into the LLM output’s correctness. Formally, an answer  $\hat{y}$  is considered correct if the conclusion extracted through a post-processing function  $\Gamma$  matches the given ground truth  $y$ . Thus, the accuracy of an LLM can be computed as

$$\mathcal{A} = \frac{1}{N} \sum_{i=1}^N \mathbb{1}(\Gamma(\hat{y}_i), y_i), \quad (2)$$

where  $N$  is the number of tested samples and  $\mathbb{1}(u, v)$  is the indicator function that returns 1 if  $u = v$ , 0 otherwise. Please note that  $\Gamma$  represents a user-defined function that can be implemented based on a regular expression (e.g., by extracting specific patterns from the sentence (Fu et al., 2023)) or using pseudo-judge approaches (e.g., by using a secondary large model as a judge (Zheng et al., 2024a)).

Starting from Equation (2), the conciseness of an output  $\hat{y}_i$  can be integrated with its correctness by multiplying the indicator function by a penalty term  $p(\hat{y}_i) \in [0, 1]$  that decreases its value for long outputs:

$$\frac{1}{N} \sum_{i=1}^N [\mathbb{1}(\Gamma(\hat{y}_i), y_i) \cdot p(\hat{y}_i)]. \quad (3)$$

The following defines three specific metrics by setting a proper penalty function.

**Hard- $k$  Concise Accuracy:** HCA( $k$ ). It measures the fraction of correct outputs that do not exceed a user-specified length  $k$ :

$$\text{HCA}(k) = \frac{1}{N} \sum_{i=1}^N [\mathbb{1}(\Gamma(\hat{y}_i), y_i) \cdot p_{\text{hard}}(\hat{y}_i, k)],$$

where

$$p_{\text{hard}}(\hat{y}_i, k) = \begin{cases} 1 & \text{if } \mathcal{N}(\hat{y}_i) \leq k \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

This metric does not account for responses that exceed the specified maximum length, thereby promoting conciseness. We believe it could be particularly useful in scenarios where strict adherence to length constraints is essential, such as in real-time systems or environments with limited computational resources.

**Soft- $k$  Concise Accuracy:**  $SCA(k, \alpha)$ . It generalizes the previous metric by penalizing the correct answers that exceed the maximum length  $k$  with a term that decreases exponentially with a decay factor  $\alpha$ :

$$SCA(k, \alpha) = \frac{1}{N} \sum_{i=1}^N [\mathbb{1}(\Gamma(\hat{y}_i), y_i) \cdot p_{soft}(\hat{y}_i, k, \alpha)],$$

where

$$p_{soft}(\hat{y}_i, k, \alpha) = \min\left(1, e^{\frac{k - \mathcal{N}(\hat{y}_i)}{-\alpha}}\right). \quad (5)$$

In the formula, the user-defined decay  $\alpha \geq 0$  can be considered a sort of tolerance that controls how much the length impacts the overall accuracy; the higher the value of  $\alpha$ , the higher the tolerance for answers exceeding the specified length  $k$ . Note that for  $\alpha = 0$ ,  $SCA(k, 0)$  reduces to  $HCA(k)$ .

**Consistent Concise Accuracy:**  $CCA(k, \alpha, \beta)$ . It further generalizes the previous metrics by also accounting for the variation in the lengths among all the outputs obtained:

$$CCA(k, \alpha, \beta) = SCA(k, \alpha) \cdot p_{var}(\sigma, \beta)$$

where

$$p_{var}(\sigma, \beta) = \min\left(1, e^{\frac{\beta - \sigma}{\beta}}\right). \quad (6)$$

In Equation (6),  $\sigma$  denotes the standard deviation of the output length distribution, whereas  $\beta$  is a parameter that controls the tolerance for having large length variations: the higher the value of  $\beta$ , the higher the tolerance. Note that, given a tolerance  $\beta$ ,  $p_{var}(\sigma, \beta) = 1$  for  $\sigma \leq \beta$ , while it decreases exponentially for  $\sigma > \beta$ .

The CCA metric aims to promote consistency in the lengths of responses, representing an important measure when the predictability of inference time is a crucial property for a specific application. A low standard deviation  $\sigma$  indicates that the model produces responses of uniform length. In contrast, a high value of  $\sigma$  denotes a model with a large response variability, making predicting its timing response time difficult.

## 5 CCoT Prompting

From the results presented in Section 3, it is clear that the relationship between output length and inference time necessitates deeper awareness. To this end, this section focuses on improving the use of

CoT, aiming to preserve the benefits of this technique while paying more attention to the length of the answers. This help achieve a better trade-off between efficiency and accuracy.

For this purpose, we introduce a constrained chain of thought (CCoT) prompt, which includes an explicit sentence to constrain the generated output to a maximum number of words, encouraging the model to compress its reasoning and produce a more concise answer in a reduced amount of time. As explained in Section 3, CoT-prompt can be computed as  $x = concat(x_{us}, x_p)$ , where  $x_p$  is an explicit request for providing reasoning steps in the generated answer (e.g., “let’s think step by step”). Technically, to push LLMs to return more concise reasoning, the CCoT-prompt is formalized as  $x = concat(x_{us}, x_p, x_l)$ , where  $x_l$  represents the sentence that specifies the constraint on the output length (e.g., “and limit the length of the answer to 30 words”). Figure 2 shows an example that illustrates the difference between a CoT and a CCoT prompt. Note that the answer generated for that specific question using a CoT prompt consists of 67 words in 17.64s, while the answer generated on the same question provided with a CCoT prompt (specifying a constraint of 45 words) consists of 34 words in 11.65s, and it is still correct.

The analysis presented in the following section and experimental part in Section 7 provide a detailed evaluation of the CCoT prompting technique using various metrics and demonstrate its benefits on the conciseness along with redundancy and information flow scores of the generated answers.

## 6 Analysis of the conciseness

To provide a deeper understanding and more comprehensive comparisons of answers in terms of conciseness, we define some conciseness properties for a given answer in the following. Intuitively, we assume that the conciseness of an answer in the CoT paradigm could be derived from the following two concepts: *how many steps are required to produce a response* and *how much information is repeated in successive steps*. While the first property is easy to derive by immediately looking at the number of steps, assuming to have a similar number of steps from two answers, extracting a numerical interpretation of the conciseness could be more complicated. For this purpose, we define two additional scores: *Redundancy*, which measures

CoT → output length 67 words; **inference time 17.64s**

Q: Tracy used a piece of wire 4 feet long to support tomato plants in the garden. The wire was cut into pieces 6 inches long. How many pieces did she obtain? **Let's think a bit step by step**

A: First, we need to convert 4 feet to inches. There are 12 inches in a foot, so 4 feet equals  $4 \times 12 = 48$  inches. Next, we can divide the total number of inches by the length of each piece (6 inches) to find the total number of pieces:  $48 \text{ inches} \div 6 \text{ inches} = 8$  pieces. Therefore, Tracy obtained 8 pieces of wire.

CCoT-45 → output length 34 words; **inference time 11.65s**

Q: Tracy used a piece of wire 4 feet long to support tomato plants in the garden. The wire was cut into pieces 6 inches long. How many pieces did she obtain? **Let's think a bit step by step and limit the answer length to 45 words.**

A: 1. Convert 4 feet to inches:  $4 \times 12 = 48$  inches 2. Divide 48 inches by 6 inches per piece:  $48 \div 6 = 8$  pieces. So, Tracy obtained 8 pieces of wire.

Figure 2: Example of CCoT in a zero-shot setting for a question extracted from the GSM8K dataset. In each box, we show the prompt and answer provided by the LLM. The box above shows the classic CoT zero-shot approach, where long reasoning is returned as output. The box below shows the use of CCoT, which reduces the number of output words while maintaining a correct answer. In the box title, we also show the word count along with inference time to clarify the improvements of the approach.

the extent to which individual steps in a generated answer repeat or overlap in *synthetical* content, and *Information Flow*, which measures how much the *semantic* information of the previous step is repeated in the current step. To compute these two scores, both CCoT and CoT answers are divided into discrete steps, based on sentence tokenization<sup>2</sup>.

## 6.1 Redundancy Score

The redundancy score for each step represents how much its content overlaps with other steps. Let a generated answer  $\hat{y}$  be divided into  $n$  steps  $S = \{s_1, s_2, \dots, s_n\}$ , where each  $s_i$  represents a single step (sentence). The redundancy mean score  $RMS(\hat{y})$  is computed as:

$$RMS(\hat{y}) = \frac{1}{n \cdot (n - 1)} \sum_i^n \sum_{j \neq i}^n \text{SyS}(s_i, s_j), \quad (7)$$

where  $\text{SyS}(s_i, s_j)$  measures the syntactical similarity between steps  $s_i$  and  $s_j$  as the ratio of matching subsequences. This is computed as  $\text{SyS}(s_i, s_j) = \frac{\text{LMS}(s_i, s_j)}{\text{TLCS}(s_i, s_j)}$ , where LMS states for *Length of Matching Subsequences* and TLCS states for *Total Length of Compared Sequences* (Ratcliff et al., 1988).

These scores help to evaluate whether the reasoning process is concise or contains redundant

synthetical repetition. We assume that for not concise answers we have a high redundancy, which indicates that steps are overly reliant on repeating information rather than advancing the reasoning.

## 6.2 Information Flow

The *Information Flow*  $\mathcal{I}(s_i, s_{i+1})$  between two consecutive steps  $s_i$  and  $s_{i+1}$  measures the extent to which the content of  $s_i$ , in a semantic sense, persists in  $s_{i+1}$ . This can be expressed as:

$$\mathcal{I}(s_i, s_{i+1}) = \text{SeS}(s_{i+1}, s_i), \quad (8)$$

where  $\text{SeS}(s_{i+1}, s_i)$  represents the semantic similarity between  $s_i$  and  $s_{i+1}$ . In our implementation, we used *BERTScore* (Zhang et al., 2019; Devlin et al., 2018) to compute the semantic similarity between steps, which leverages contextual embeddings generated by BERT. For the last step, since there is no subsequent step, we assume  $\mathcal{I}(s_n) = 0$ .

The rationale behind this score is that, when the number of steps is similar, the information flow across the steps tends to be higher in less concise answers because each step relies on a more similar set of details and context from the previous one, increasing the risk of repetitiveness.

## 7 Experiments

This section presents a set of experiments carried out to evaluate the effectiveness of the proposed CCoT approach under classic metrics, as well as

<sup>2</sup><https://www.nltk.org/api/nltk.tokenize.tokenize.html>