Information Sciences 757 (2026) 123939

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Concise thoughts: Impact of output length on LLM reasoning and cost

Sania Nayab[a,*], Giulio Rossolini[a], Marco Simoni[b,c], Andrea Saracino[a,c], Giorgio Buttazzo[a], Nicolamaria Manes[d], Fabrizio Giacomelli[d]

[a] Department of Excellence in Robotics and AI, Scuola Superiore Sant'Anna, Pisa, Italy
[b] Sapienza Università di Roma, Rome, Italy
[c] Institute of Informatics and Telematics, National Research Council of Italy, Pisa, Italy
[d] Mediavoice Srl - Roma e Napoli, Roma, Italy

HIGHLIGHTS

- Introduced Constrained-CoT (CCoT), a prompt engineering method to limit LLM answer length, improving the trade-off between generation time and concise correctness.
- Proposed three novel metrics: HCA, SCA, and CCA to evaluate correctness while considering the conciseness of LLM outputs.
- Analyzed conciseness via redundancy and information flow to understand effects of output length constraints.
- Conducted experiments across datasets and LLMs, showing that CCoT improves inference time, accuracy, and conciseness compared to the original CoT.
- Demonstrated advantages of the proposed metrics in effectively assessing conciseness.

ARTICLE INFO

Keywords:
Large language models
Prompt engineering
Chain-of-thought
LLMs metrics

ABSTRACT

Today's large language models (LLMs) can solve challenging question-answering tasks, and prompt engineering techniques, such as chain-of-thought (CoT), have emerged as prominent methods for enhancing output reasoning and accuracy. However, many models and techniques tend to produce excessively verbose answers, presenting challenges for both computational efficiency and conciseness. To address this, this paper analyzes the impact of output lengths on LLM inference pipelines by introducing and proposing novel metrics to evaluate the *correct conciseness* of a model and related prompting techniques. Then, we examine the impact of controlling output length through a refined prompt engineering strategy, Constrained-CoT (CCoT), which encourages the model to produce more concise outputs. To better understand the effects of such prompting constraints, we also introduce two additional metrics to analyze conciseness, measured in terms of redundancy and information flow in generated answers. Experiments conducted on various LLMs and datasets demonstrate the benefits of the proposed metrics and the robustness of CCoT across different architectures.

* Corresponding author.
Email addresses: sania.nayab@santannapisa.it (S. Nayab), giulio.rossolini@santannapisa.it (G. Rossolini), marco.simoni@iit.cnr.it (M. Simoni), andrea.saracino@santannapisa.it (A. Saracino), giorgio.buttazzo@santannapisa.it (G. Buttazzo), manes@mediavoice.it (N. Manes), giacomelli@mediavoice.it (F. Giacomelli).

https://doi.org/10.1016/j.ins.2026.123939
Received 26 May 2025; Received in revised form 15 July 2026; Accepted 22 July 2026
Available online 24 July 2026
0020-0255/© 2026 Elsevier Inc. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

1. Introduction

In recent years, large language models (LLMs) have demonstrated remarkable capabilities in tackling complex question-answering tasks, making significant strides in natural language understanding and generative AI [1–4]. The continuous advancements made in architectures and training methods have played a crucial role in enhancing the performance of these models. Alongside these improvements, prompt engineering has emerged as a key technique to further boost LLM performance, particularly Chain-of-Thought (CoT) prompting [5,6], which helps improve output correctness by encouraging the model to articulate intermediate reasoning steps.

Despite its advantages, the CoT prompting can lead to long outputs, increasing the time required for the model to generate a response. This is due to the nature of autoregressive transformers, which decode text word by word [7,8], which implies that the time required to generate a response could be unbounded and heavily influenced by the length of the reasoning provided. Such lengthy and variable response delays can be undesirable when an LLM interacts with a user in a conversational setting. These limitations highlight the need to consider: *(i)* evaluation metrics that jointly assess the conciseness and correctness of model outputs, and *(ii)* methods to prevent excessively long chains of reasoning while preserving response quality.

In this work, we address these challenges by studying how conciseness can be encouraged through prompt design and by analyzing how models behave when concise reasoning is enforced. Specifically, we investigate model behavior both in the output space, through dedicated evaluation metrics for concise correctness, and in the embedding space through analyses of redundancy and information flow within the reasoning process. The first part of this work emphasizes the importance of accounting for answer length in evaluating correctness as an indicator of computational cost. This is achieved by introducing three novel metrics (HCA, SCA, and CCA) that assess both the brevity and correctness of a generated answer. The proposed metrics aim to reweight a model's accuracy by considering output-length aspects that may affect inference time and time predictability.

To address the second issue, this work explores how to retain the benefits of CoT reasoning while gaining better control over the length of the generated reasoning through prompt design. To this end, we introduce a refined prompting strategy called Constrained Chain-of-Thought (CCoT), which encourages LLMs to generate more concise outputs by explicitly constraining the reasoning length. To address the second issue, this work explores how to retain the benefits of CoT reasoning while gaining better control over the length of the generated reasoning through prompt design. To this end, we introduce a refined prompting strategy called Constrained Chain-of-Thought (CCoT), which encourages LLMs to generate more concise outputs by explicitly constraining the reasoning length. The core idea of this approach is to allow users to specify a flexible length constraint that acts as a tunable parameter, thereby enabling control over the degree of conciseness in the generated answers. This mechanism aims to better support controlled reasoning from a prompting perspective and is used in this work to further investigate model behavior when attempting to balance reasoning quality and conciseness while preserving answer correctness.

Finally, to better assess the ability of LLMs to follow such constraints and to gain deeper insights into conciseness within the model's embedding space, we introduce additional analysis scores that measure redundancy and information flow in the generated reasoning.

To summarize, this work provides the following main contributions:

- Three novel metrics (HCA, SCA, and CCA) to evaluate LLM outputs by considering both correctness and conciseness.
- Introduction of the concept of Constrained-CoT, a prompt engineering strategy designed to limit the length of answers generated by LLMs, thereby enhancing the trade-off between generation time and conciseness.
- Introduction of an analysis of the conciseness in terms of redundancy and information flow for a given answer, thus offering an understanding of the effects of constraining output length.
- We conducted multiple experiments to analyze the impact of prompts and the proposed metrics across different datasets and LLMs, demonstrating benefits in terms of inference time, accuracy and conciseness.

The rest of the paper is organized as follows. Section 2 discusses the literature related to this work. Section 3 motivates the study, presents a set of metrics for conciseness, and introduces the proposed CCoT approach, along with an analysis for evaluating the conciseness of a given answer. Section 4 reports the results of experiments conducted on pre-trained models using three arithmetic reasoning datasets. Section 5 discusses the limitations of the approach and outlines future research directions. Finally, Section 6 presents the conclusions. The code is publicly available.[1]

2. Related work

This section highlights challenges in LLM conciseness, where accuracy-focused methods increase verbosity and computational cost, and examines how prompt engineering, including CoT, exacerbates verbosity and inference time.

LLM accuracy and verbosity. Recent works on LLMs have primarily focused on improving their accuracy [9–11]. However, as models scale up, they tend to generate more extensive and articulated responses [12,13], which can introduce problems such as hallucinations (where the model produces information that appears plausible but is not grounded [14]) or unnecessarily long explanations [15,16]. These longer responses can obscure key information, making it difficult for users to extract relevant content efficiently [17,18].

[1] https://github.com/sanianayab/Constrained-Chain-of-Thought-CCoT-.

Prompt engineering and reasoning. Several techniques have been proposed to address these challenges, such as multi-hop processing [19], which extracts relevant rationale from an input prompt for a given question. However, these methods do not directly tackle the underlying issue of balancing accuracy with conciseness, nor do they offer a means of controlling response length without sacrificing information quality.

In the domain of reasoning, CoT prompting [5] has shown significant benefits for tasks like question answering by requiring the model to provide a step-by-step explanation. However, as highlighted in Section 3.1, CoT prompting tends to generate lengthy answers, which increases both the generation time and the risk of verbosity [20,21].

Conciseness in LLM evaluation. Despite recent advancements, several key aspects remain insufficiently explored: *(i)* the impact of concise answers on inference cost and time predictability; *(ii)* the integration of such aspects into unified metrics that evaluate LLMs not only in terms of correctness but also conciseness; and *(iii)* an understanding of the analysis of conciseness based on the embeddings content extracted by the generated answers.

To address these challenges, our work introduces novel metrics that jointly account for the conciseness and correctness of generated responses. Additionally, two new scores are proposed to assess conciseness from the embeddings produced by the model. These scores focus on the importance of reasoning steps by analyzing redundancy and information flow. Then, to evaluate the ability of LLMs to control the length of reasoning in their outputs, this work studies a refined version of the CoT prompting [5], termed Constrained-CoT. This approach explicitly guides the model to limit the length of its reasoning while preserving the quality of its answers and improving the inference time. This is achieved by improving the conciseness of the responses, which is analyzed and assessed using the proposed scores.

3. Methodology

In this section, we first explore the motivations for focusing on concise outputs, highlighting preliminary experiments that demonstrate the need for improved efficiency in reasoning tasks. We then delve into the proposed CCoT prompting method, explaining its steps and providing illustrative examples. Next, we discuss the proposed metrics in a structured manner, emphasizing their role in jointly evaluating the correctness and conciseness of generated responses. Finally, we present an in-depth analysis of conciseness, examining how additional embedding-level metrics can be used to quantify and control output length, redundancy, and information flow in the reasoning process.

3.1. Motivations

The output generation time of an LLM depends on various factors, including the model architecture, the pre-and post-processing steps, the answer decoding process, and the question posed, also considering the use of prompt engineering approaches. While the computational cost due to the architecture is well understood, the influence of the other aspects on the overall generation time is less clear and requires further investigation. More formally, an LLM can be represented as a function f that takes as input a prompt x with $\mathcal{N}(x)$ tokens and generates an output $\hat{y} = f(x)$, having $\mathcal{N}(\hat{y})$ tokens, where $\mathcal{N}$ is a length operator that simply counts the number of tokens. The input x can be considered as composed of the original user input x_{us} and a prompt engineering text x_p, depending on the technique used. For instance, in a zero-shot CoT setting, the prompt can be computed as $x = concat(x_{\text{us}}, x_p)$, where x_p is an explicit request for providing reasoning steps in the answer and $concat(a, b)$ is the concatenation operator that merges two vectors a and b into a single one.

In an encoder–decoder architecture, such as the one used by Transformers [7], let $f_e(x)$ and $f_d(x)$ denote the functions associated with the encoder and the decoder, respectively. Then, the output $\hat{y}$ is a list of tokens $[a^{(1)}, \dots, a^{(\mathcal{N}(\hat{y}))}]$, where each $a^{(i)}$ is computed based on the previously generated tokens and the encoder's embedding representation $f_e(x)$. That is,

$$a^{(i)} = f_d(f_e(x), [a^{(0)}, \dots, a^{(i-1)}]), \quad i > 0. \tag{1}$$

From Eq. (1), it is clear that the larger the set of output tokens in the answer, the longer the time the model takes to generate the answer due to the increased number of times the decoder is invoked. The same consideration could also be 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 [22]. The results of this test are illustrated in Fig. 1(a), where red and blue dots refer to answers given with and without CoT, respectively.

The scatter plot shows that CoT significantly increases output length and generation time. In these plots, we analyze the output length in terms of words. However, as is well known in the literature, the relationship between words and tokens can be easily derived.[2] This suggests that while CoT improves the correctness of responses (see Section 4), more attention should be given to the time cost it introduces. To better appreciate the impact of CoT on the output length, Fig. 1(b) 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 regions where the two bars overlap.

[2] To further illustrate this relationship, we provide additional analyses highlighting the mapping between the number of generated words and tokens (see Fig. A.7 in Appendix A.1).

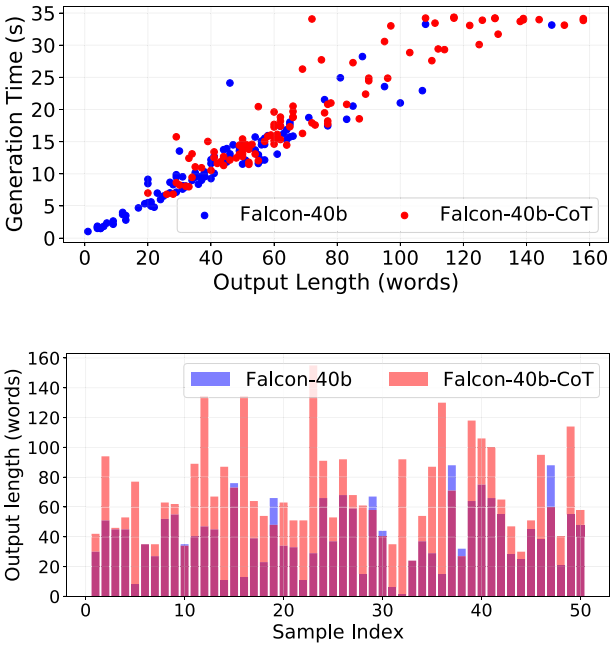


Fig. 1. Analysis of the impact of CoT on Falcon-40b efficiency: (top) Relation between response time and output length, without CoT (blue dots) and with CoT (red dots), across 100 questions from the GSM8K test set. (bottom) output words variation between output length with CoT and without CoT using 50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

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 × 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 inches ÷ 6 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 × 12 = 48 inches 2. Divide 48 inches by 6 inches per piece: 48 ÷ 6 = 8 pieces. So, Tracy obtained 8 pieces of wire.

Fig. 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 display the word count along with inference time to highlight the improvements of the approach.

3.2. *CCoT prompting*

From the results presented in Section 3.1, 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 helps 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 3.1, CoT-prompt can be computed as $x = concat(x_{\text{us}}, x_p)$, 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 $x = concat(x_{\text{us}}, x_p, x_l)$, 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"*). Fig. 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 for 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 approach presented in the Section 3.4 section and experimental part in Section 4 provide a detailed evaluation of the CCoT prompting technique using various metrics and demonstrate its benefits in the of conciseness as well as redundancy and information flow scores of the generated answers.

3.3. Metrics for correct conciseness

Motivated by the previous considerations, this section presents three novel metrics to evaluate at testing time 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 4.3, impact 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 Γ matches the given ground truth y [5,6,23]. Thus, the accuracy of an LLM can be computed as

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

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 Γ represents a user-defined function that can be implemented based on a regular expression (e.g., by extracting specific patterns from the sentence [5,6]). Starting from Eq. (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:

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

The following defines three correct conciseness metrics by setting a proper penalty function. In this study, we use three user-defined parameters, k, α, and β, to balance conciseness and correctness in the generated responses. These parameters are essential for tailoring the model's behavior to different tasks and use cases, allowing flexibility in controlling output length, reasoning depth, and tolerance for variability.

Hard-k concise accuracy. HCA(k). It measures the fraction of correct outputs that do not exceed a user-specified length k:

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

where

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

Here, $\mathcal{N}(\hat{y}_i)$ denotes the number of words in the generated answer $\hat{y}_i$. $p_{hard}(\hat{y}_i, k)$ is a penalty function for hard conciseness enforcement. It returns 1 if the length of $\hat{y}_i$ is less than or equal to k, otherwise it returns 0. To conclude, 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, α). It generalizes the previous metric by penalizing correct answers that exceed the maximum length k with a term that decreases exponentially with a decay factor α, allowing for a smooth tolerance for longer outputs. Specifically, the penalty decreases as the length exceeds k, and the decay factor α controls the amount of tolerance:

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

where

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

In this formula, $p_{soft}(\hat{y}_i, k, \alpha)$: The soft penalty function applies an exponential penalty for outputs longer than k, based on the decay factor α. The user-defined decay $\alpha \geq 0$ serves as a tolerance factor that controls how much the output length affects the overall accuracy. The larger the value of α, the higher the tolerance for outputs that exceed the specified length k. When $\alpha = 0$, SCA reduces to HCA, i.e., $\text{SCA}(k, 0) = \text{HCA}(k)$, representing a hard constraint where any output exceeding the length is heavily penalized.

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

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

In Eq. (6), σ denotes the standard deviation of the output length distribution, whereas β is a parameter that controls the tolerance for having large length variations: the higher the value of β, the higher the tolerance. Note that, given a tolerance β, $p_{var}(\sigma, \beta) = 1$ 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. It uses the global standard deviation σ of output lengths to capture workload-level predictability. A low standard deviation σ indicates that the model produces responses of uniform length. In contrast, a high value of σ denotes a model with a large response variability, making predicting its timing response time difficult.

3.4. 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 can 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 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 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.[3]

3.4.1. 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 $S = \{s_1, s_2, \ldots, s_n\}$, where each s_i represents a single step (sentence). The redundancy mean score $RMS(\hat{y})$ is computed as:

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

where $\mathrm{SyS}(s_i, s_j)$ measures the syntactical similarity between steps s_i and s_j as the ratio of matching subsequences. This is computed as $\mathrm{SyS}(s_i, s_j) = \frac{\mathrm{LMS}(s_i, s_j)}{\mathrm{TLCS}(s_i, s_j)}$, where LMS stands for *Length of Matching Subsequences* and TLCS stands for *Total Length of Compared Sequences* [24].

These scores help to evaluate whether the reasoning process is concise or contains redundant synthetical repetition. We assume that for non-concise answers we have a high redundancy, which indicates that steps are overly reliant on repeating information rather than advancing the reasoning.

3.4.2. 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:

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

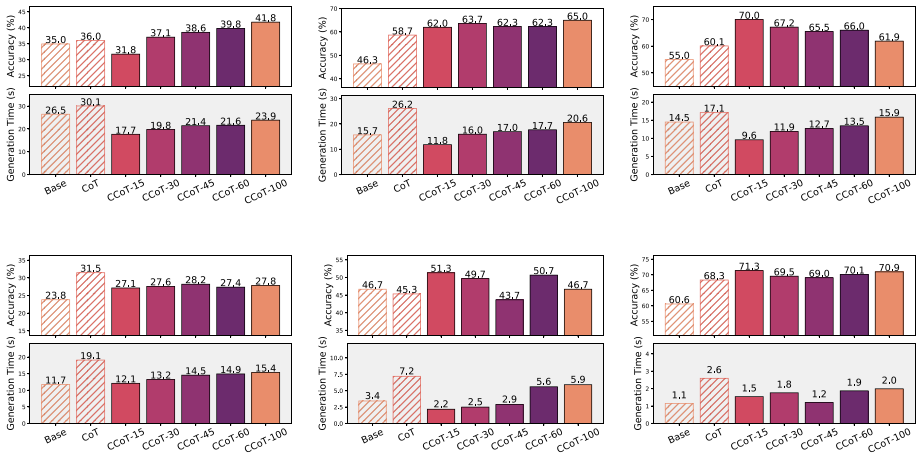
where $SeS(s_{i+1}, s_i)$ represents the semantic similarity between s_i and s_{i+1}. In our implementation, we used *BERTScore* [25,26] to compute the semantic similarity between steps, which leverages contextual embeddings generated by BERT. For the last step, since there is no subsequent step, we 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.

4. Experimental results

This section presents a set of experiments carried out to evaluate the effectiveness of the proposed CCoT approach under classic metrics, as well as to illustrate the benefits of the proposed metrics and scores in highlighting the trade-off between accuracy and computational cost. Specifically, the following research questions are investigated in the next experiments: **RQ1** Is the CCoT approach beneficial in terms of efficiency and accuracy compared to classic CoT?;**RQ2** Are the proposed metrics representative of showing this trade-off?; and **RQ3** Are the CCoT answers actually more concise, and to what extent are LLMs capable of controlling the output length based on an explicit prompt request?

[3] https://www.nltk.org/api/nltk.tokenize.sent_tokenize.html.



(a) GSM8k (b) SVAMP (c) ASDIV

Fig. 3. Accuracy (white plots, the higher the better) and mean generation time (background colored plots, the lower the better) for Llama2-70b (top row) and Falcon-40b (bottom row) on the GSM8K, SVAMP, and ASDIV datasets. The models are evaluated using plain prompts (base), CoT, and CCoT under different length constraints.

4.1. Experimental setup

All the experiments have been carried out with the Text Generation Inference (TGI) platform[4] on 8 NVIDIA A100 GPUs.

Models. Specifically, we evaluated large and open source pre-trained LLMs from Hugging Face,[5] such as instruction-tuned models Falcon-40b-instruct and models trained, and reinforced by utilizing private data, namely Llama2-70b-chat-hf [27]. Additionally, further experiments on smaller models were conducted to evaluate their capability in handling CCoT, with detailed results provided in Appendix Appendix B.

Datasets. The experiments utilized three arithmetic reasoning datasets: GSM8k [22], SVAMP [28], and ASDIV [29], commonly used to assess models' mathematical inference and computational reasoning abilities. The GSM8k test set includes over 1.3k problems out of 8000. The SVAMP test set contains 300 examples, and the ASDIV test set comprises 1.22k examples.

For all experiments, the effectiveness of CCoT was compared by assessing the selected LLMs both with and without CoT (base mode).

4.2. Impact of CCoT on accuracy and efficiency

This experiment was carried out to evaluate the impact of *CCoT* on computation time and accuracy (**RQ1**). In particular, the selected LLMs were evaluated on three datasets using plain prompts (base), *CoT*, and *CCoT* with different length constraints:15, 30, 45, 60, 100. The results are presented in Fig. 3 for Llama2-70b (top) and for Falcon-40b (bottom), showing accuracy and generation time in the first and second rows, respectively.

Considering the results with Llama2-70b, CCoT prompting demonstrates the ability to reduce generation time compared to CoT and, in most cases, achieves a time reduction similar to or better than plain prompting (base). Additionally, it generally improves or minimally impacts accuracy, thereby enhancing the trade-off in both directions.

For instance, under the considered hardware settings, and as an illustrative indication,[6] the average generation time decreases from 30.1 s with CoT to 23.86 s with CCoT on GSM8K using Llama2-70B with a length constraint of 100. The generation time decreases further under stricter constraints (smaller k).

At the same time, the accuracy consistently improves; for example, with the GSM8k dataset, the accuracy of Llama2-70b increases from 36% with CoT to 37% (with CCoT-30) and 41.77% (with CCoT-100).

Similar observations can be made for the results with Falcon-40b (bottom part), where the CCoT approach improves the trade-off between efficiency and accuracy across all three datasets. In particular, CCoT achieves better accuracy overall for SVAMP and ASDIV, while significantly reducing generation time. This improvement in terms of accuracy, however, does not occur for the GSM8K dataset, where the sentences are more complex than those in the other two datasets. We believe that such a complexity makes it more challenging for a medium-sized model like Falcon-40b to effectively handle the CCoT constraint. Nonetheless, the accuracy remains higher than the base model, indicating that the CoT-based approach still provides benefits.

[4] https://huggingface.co/docs/text-generation-inference.

[5] https://huggingface.co/blog/os-llms.

[6] Inference time, based on the used settings and hardware configurations, is reported only as an illustrative practical effect of token-generation reduction.

We also acknowledge that different behaviors may arise when dealing with datasets of varying complexity, as the nature of the questions can differ. For instance, CCoT-15 outperforms other CCoT variations because its 15-word responses potentially align better with the simpler nature of this dataset. We argue that this effect is closely related to the complexity of the questions, which makes the constraint impactful even on accuracy.

4.3. Analysis of the correct conciseness

Based on the previous results, CCoT demonstrates clear benefits in balancing accuracy and computational efficiency. To unify these aspects, we introduced new metrics in Section 3.3 that integrate answer length with correctness. The analysis presented in Tables 1 and 2 for Llama2-70b and Falcon-40b, respectively, highlights how these metrics effectively combine cost and accuracy (**RQ2**), while also remarking on the advantages of CCoT. We also added, at the bottom of the table (highlighted in gray), the reference accuracy metric to facilitate a comparative analysis of the performance drop when using the proposed metrics.

HCA evaluation. The *Hard-k concise accuracy* evaluates the accuracy considering only the correct answers whose length is less than a specified value k. The top parts of Tables 1 and 2 report the value of this performance index across the three datasets, when using the different prompt approaches and for different values of k.

Specifically, for both Llama2-70b and Falcon-40b, the use of CCoT gets better results compared to base and CoT prompts across all values of k. Notably, for lower values of k, CoT prompts exhibit a significant reduction in performance, while this accuracy drop can be mitigated by using CCoT with strict length constraints, such as 15 or 30.

SCA evaluation. We also evaluated both models using the *Soft Conciseness Accuracy (SCA)*, across different k values and α, where α represents a tolerance for accepting answers longer than the desired limit k. This metric is a generalization of the *HCA*, giving more flexibility in considering correct answers that are longer but still close to the desired length k.

The SCA values computed for Llama2-70b and Falcon-40b on the datasets are reported in center parts of the tables for different values of k and a fixed tolerance value $\alpha = 10$. For both models, the SCA values in CCoT settings are often comparable to HCA values for high values of k, such as 80 or 100. This is because, as discussed in Section 4.5, for such lengths, the CCoT prompts are effective at returning outputs below the desired limit, making the tolerance less necessary. Conversely, for smaller k values, such as $k = 40$, SCA starts exceeding HCA, indicating that some correct answers have a length larger than k. However, for such k values of, using a tolerance α results in more pronounced improvements for CCoT prompts compared to Base and CoT. This means that, although many correct outputs are longer than k, under CCoT the model is still encouraged to constrain them close to k, thus achieving a higher score. This effect is particularly noticeable on Llama2-70b, which is more capable of controlling the length and produce correct outputs than Falcon-40b.

CCA evaluation. The *Consistent Concise Accuracy* measures the capability of a model to generate correct answers whose lengths do not vary significantly, and therefore are consistent with the specified constraint. The *CCA* requires a third parameter β (in addition to k and α), denoting a tolerance for the output length variability.

The bottom parts of the tables report the CCA scores obtained on Llama2-70b and Falcon-40b for $\alpha = 10$, $\beta = 40$, and different values of k, for the various prompting methods. According to these settings, the CCoTs result in a clear improvement compared to CoT prompting and base, for both Llama2-70b and Falcon-40b. However, for high CCoT length constraints (e.g., 100), the CCA score tends to decrease, which does not happen with the other two metrics. This can be explained by considering that an increased length constraint gives the model more freedom to generate outputs with higher variations, as discussed in Section 4.5.

The robustness of the proposed metrics hinges on the careful selection of user-defined parameters (k, α, and β), which provide the flexibility to adjust conciseness and correctness across different applications. The ability to vary these parameters ensures that the model can be analysed for different constrained lengths to suit specific tasks, balancing efficiency and accuracy.

4.4. Understanding the effect of CCoT

While the previous experiments demonstrate that CCoT achieves a clear reduction in the number of words produced, thus improving the trade-off between accuracy and generation time, the scores introduced in Section 3.4 are used in the following analysis to quantify the benefits in terms of conciseness (**RQ3**). As detailed in Section 3.4, the conciseness of an answer is assumed to depend on two key properties: *(i)* the number of steps required to produce a response, and *(ii)* assuming a similar number of steps, the amount of information repeated across those steps.

Number of steps. To evaluate the first property, we plotted in Fig. 4 the distribution of the percentage of generated answers based on the *number of reasoning steps* [30] for Llama2-70b (top row) and Falcon-40b (bottom row) across the three datasets.

The results reveal a significantly different distribution in the number of steps across all datasets with Falcon-40b and for the ASDIV dataset with Llama2-70b. In fact, in these cases, the CCoT curves are clearly concentrated toward a reduced number of steps compared to CoT, indicating that the CCoT prompt consistently generates responses with fewer steps.

While the analysis of the number of steps explains the improved conciseness achieved with CCoT, more detailed analysis of the redundancy and information flow is required to understand the enhanced conciseness observed with Llama2-70b on GSM8K and SVAMP, where the number of steps follows a trend similar to that of CoT.

Table 1
Llama2-70b.

	Base	CoT	CCoT15	CCoT30	CCoT45	CCoT60	CCoT100
GSM8K - HCA							
H-∞	35.0	36.0	31.8	37.1	38.6	39.8	**41.8**
H-100	29.9	22.9	31.2	35.3	37.5	38.7	**38.9**
H-80	22.0	15.4	29.2	31.8	33.1	**35.0**	31.6
H-40	4.8	0.8	**12.7**	10.8	8.0	8.5	4.5
SVAMP - HCA							
H-∞	46.3	58.7	62.0	63.7	62.3	62.3	**65.0**
H-100	46.3	50.0	59.7	61.0	61.0	59.7	**61.7**
H-80	46.3	41.0	56.7	**57.7**	57.0	54.3	54.7
H-40	23.3	12.0	**44.7**	34.0	30.0	30.0	20.0
ASDIV - HCA							
H-∞	52.8	60.3	**67.2**	65.6	64.3	65.0	61.4
H-100	52.8	60.3	**67.2**	65.6	64.3	65.0	61.4
H-80	52.8	60.3	**67.2**	65.6	64.3	65.0	61.4
H-40	31.7	23.6	**55.5**	46.4	44.4	43.8	29.5
GSM8K - SCA ($\alpha = 10$)							
SCA-100	31.4	26.9	31.5	36.3	38.1	39.2	**40.1**
SCA-80	25.6	19.0	30.1	33.8	35.3	**36.6**	35.0
SCA-40	8.5	3.3	**18.0**	16.9	15.4	16.3	11.0
SVAMP - SCA ($\alpha = 10$)							
SCA-100	46.3	52.3	60.3	61.6	61.3	60.6	**62.7**
SCA-80	46.3	45.6	58.4	**59.2**	59.1	57.6	58.1
SCA-40	31.4	19.5	**48.2**	41.1	39.6	38.7	31.6
ASDIV - SCA ($\alpha = 10$)							
SCA-100	52.8	60.3	**67.2**	65.6	64.3	65.0	61.4
SCA-80	52.8	60.3	**67.2**	65.6	64.3	65.0	61.4
SCA-40	39.5	35.2	**60.4**	54.0	52.1	51.7	40.9
GSM8K - CCA ($\alpha = 10$, $\beta = 40$)							
CCA-100	31.4	26.9	31.5	36.3	38.1	39.2	**40.1**
CCA-80	25.6	19.0	30.1	33.8	35.3	**36.6**	35.0
CCA-40	8.5	3.3	**18.0**	16.9	15.4	16.3	11.0
SVAMP - CCA ($\alpha = 10$, $\beta = 40$)							
CCA-100	46.3	52.3	60.3	61.6	61.3	60.6	**62.7**
CCA-80	46.3	45.6	58.4	**59.2**	59.1	57.6	58.1
CCA-40	31.4	19.5	**48.2**	41.1	39.6	38.7	31.6
ASDIV - CCA ($\alpha = 10$, $\beta = 40$)							
CCA-100	52.8	60.3	**67.2**	65.6	64.3	65.0	61.4
CCA-80	52.8	60.3	**67.2**	65.6	64.3	65.0	61.4
CCA-40	39.5	35.2	**60.4**	54.0	52.1	51.7	40.9
Accuracy							
GSM8K	35.0	36.0	31.8	37.1	38.6	39.8	**41.8**
SVAMP	46.3	58.7	62.0	63.7	62.3	62.3	**65.0**
ASDIV	55.0	60.1	**70.0**	67.2	65.5	66.0	61.9

Redundancy evaluation. As discussed in the previous paragraph, the distribution of reasoning steps for Llama2-70b on the GSM8K and SVAMP datasets (Figs. 4(a) and (b)) reveals that the quartile ranges of the answer distributions for CCoT and CoT answers are very similar. Specifically, in both datasets, the interquartile range spans approximately 4 to 12 steps (Q1 to Q3) for SVAMP and around 5 to 14 steps for GSM8K. This suggests comparable step counts across CCoT and CoT answers, within such ranges of reasoning steps.

To evaluate the conciseness of these reasoning steps more effectively, we present redundancy scores in Fig. 5. This figure highlights the differences in redundancy between CCoT and CoT answers, categorized by their number of steps. Notably, when focusing on step intervals within the interquartile range (Q1 to Q3, where most answers fall), which are highlighted in grey in the plots, redundancy scores are consistently higher for CoT than for CCoT. This demonstrates that CCoT achieves improved syntactic conciseness compared to CoT, even when the number of steps is similar.

Table 2
Falcon-40b.

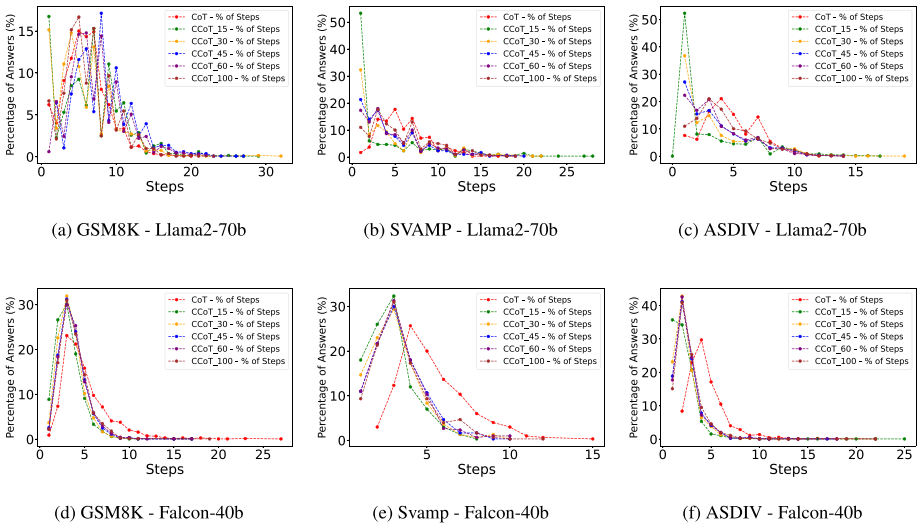
	Base	CoT	CCoT15	CCoT30	CCoT45	CCoT60	CCoT100
GSM8K - HCA							
H-∞	23.8	31.5	27.1	27.6	**28.2**	27.4	27.8
H-100	23.7	**29.3**	26.7	27.3	27.2	26.8	27.4
H-80	22.8	**26.8**	25.8	26.1	25.9	25.2	26.0
H-40	13.0	10.4	**13.4**	**13.4**	12.2	11.8	12.5
SVAMP - HCA							
H-∞	46.7	45.3	**51.3**	49.7	43.7	50.7	46.7
H-100	46.7	42.3	**51.3**	49.3	43.0	48.3	45.3
H-80	46.3	39.7	**49.7**	47.3	40.7	46.7	44.3
H-40	**38.7**	20.3	36.7	36.0	31.7	33.0	32.7
ASDIV - HCA							
H-∞	60.6	68.3	**71.3**	69.5	69.0	70.1	70.9
H-100	60.6	67.1	**71.2**	69.1	68.8	69.7	70.1
H-80	60.1	65.2	**70.9**	68.5	68.1	68.6	69.0
H-40	57.2	43.3	**64.9**	59.6	57.3	58.3	56.7
GSM8K - SCA ($\alpha = 10$)							
SCA-100	23.7	**30.1**	26.8	27.4	27.5	27.0	27.5
SCA-80	23.3	**28.1**	26.2	26.7	26.6	26.0	26.6
SCA-40	16.6	15.3	**18.0**	17.9	16.8	16.5	16.7
SVAMP - SCA ($\alpha = 10$)							
SCA-100	46.7	43.3	**51.3**	49.4	43.1	48.9	45.4
SCA-80	46.6	41.2	**50.7**	48.1	41.6	47.7	44.9
SCA-40	40.6	27.8	**42.7**	40.8	34.8	38.0	35.8
ASDIV - SCA ($\alpha = 10$)							
SCA-100	**84.4**	67.3	71.2	69.3	68.9	69.9	70.4
SCA-80)	**81.1**	66.1	71.0	68.8	68.5	69.2	69.5
SCA-40	66.0	50.4	**67.3**	63.3	61.5	61.9	61.2
GSM8K - CCA ($\alpha = 10$, $\beta = 40$)							
CCA-100	23.7	**30.1**	26.8	27.4	27.5	27.0	27.5
CCA-80	23.3	**28.1**	26.2	26.7	26.6	26.0	26.6
CCA-40	16.6	15.3	**18.0**	17.9	16.8	16.5	16.7
SVAMP - CCA ($\alpha = 10$, $\beta = 40$)							
CCA-100	46.7	43.3	**51.3**	49.4	43.1	48.9	45.4
CCA-80	46.6	41.2	**50.7**	48.1	41.6	47.7	44.9
CCA-40	40.6	27.8	**42.7**	40.8	34.8	38.0	35.8
ASDIV - CCA ($\alpha = 10$, $\beta = 40$)							
CCA-100	**84.4**	67.3	71.2	69.3	68.9	69.9	70.4
CCA-80	**81.1**	66.1	71.0	68.8	68.5	69.2	69.5
CCA-40	**66.0**	50.4	67.3	63.3	61.5	61.9	61.2
Accuracy							
GSM8K	23.8	**31.5**	27.1	27.6	28.2	27.4	27.8
SVAMP	46.7	45.3	**51.3**	49.7	43.7	50.7	46.7
ASDIV	60.6	68.3	**71.3**	69.5	69.0	70.1	70.9

To better quantify the reduction in redundancy achieved by CCoT, we define the *Mean Redundancy Reduction (MRR)* across single-step redundancies RMS_i for all reasoning steps i, and the *Overall Redundancy Reduction (ORR)* as:

$$\text{MRR} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\text{CoT RMS}_i - \text{CCoT RMS}_i}{\text{CoT RMS}_i} \times 100\right)$$

$$\text{ORR} = \frac{\text{CoT RMS} - \text{CCoT RMS}}{\text{CoT RMS}} \times 100$$

Table 3 presents the average ORR and MRR calculated for the SVAMP and GSM8K datasets, focusing on steps within the interquartile range (Q1 to Q3). The results indicate that CCoT consistently reduces redundancy. For SVAMP, the redundancy reduction ranges from 19.77% to 22.72% (ORR), while for GSM8K, it ranges from 12.64% to 24.74% (ORR).



(a) GSM8K - Llama2-70b (b) SVAMP - Llama2-70b (c) ASDIV - Llama2-70b

(d) GSM8K - Falcon-40b (e) Svamp - Falcon-40b (f) ASDIV - Falcon-40b

Fig. 4. Percentage of number of generated answers as per 'no. of reasoning steps' of Llama2-70b (top row) and Falcon-40b (bottom row) on the GSM8K (a), SVAMP (b), and ASDIV (c) test datasets.

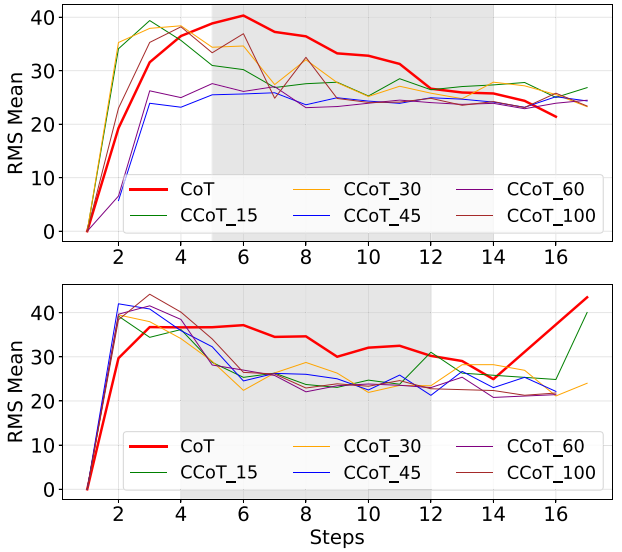


Fig. 5. RMS score (lower values indicate better conciseness) of Llama2-70b on the GSM8K (top) and SVAMP (bottom) datasets. The grey area highlights the portion of the answer distribution between the Q1 and Q3 quartiles.

Table 3
Mean and Overall Redundancy Reduction for SVAMP and GSM8K Datasets.

Method	Dataset	CCoT15	CCoT30	CCoT45	CCoT60	CCoT100
MRR	SVAMP	20.02	22.50	21.48	22.92	20.14
	GSM8K	13.58	11.65	22.81	23.23	16.33
ORR	SVAMP	20.22	22.57	21.29	22.72	19.77
	GSM8K	15.34	12.64	24.62	24.74	16.81

Information flow evaluation. To demonstrate an improved level of conciseness also from a semantic perspective, we analyze the information flow. Specifically, the median number of steps in the answer distributions for CoT and CCoT (Figs. 4(a) and (b)) is approximately 8 for both the SVAMP and GSM8K datasets. Thus, focusing on answers with a total of 8 steps, we present in Tables 4 the *Information Flow* between consecutive steps $i \rightarrow j$ for Llama2-70b on the GSM8K (top) and SVAMP (bottom) datasets.

In Table 4, CCoT-15 exhibits the largest reductions across all steps, for instance, ranging from 26% to 41% compared to CoT scores for GSM8K, while CCoT-100 shows the smallest reductions (4% to 20%), preserving more redundant semantic information. Generally,

Table 4
Information flow mean values comparison for GSM8K (top) and SVAMP (bottom) across answers with 8 steps. Better information flow indicates lower semantic conciseness between steps (highlighted with blue-like colors in the tables).

GSM8K - Llama2-70b

Steps	CoT	CCoT-15	CCoT-30	CCoT-45	CCoT-60	CCoT-100
1 ⇒ 2	0.5287	0.3417	0.39	0.49	0.49	0.43
2 ⇒ 3	0.59	0.39	0.45	0.31	0.32	0.53
3 ⇒ 4	0.56	0.37	0.43	0.36	0.37	0.47
4 ⇒ 5	0.55	0.38	0.41	0.36	0.36	0.46
5 ⇒ 6	0.56	0.39	0.42	0.36	0.37	0.51
6 ⇒ 7	0.56	0.42	0.45	0.37	0.37	0.54
7 ⇒ 8	0.52	0.42	0.47	0.57	0.56	0.50

SVAMP - Llama2-70b

Steps	CoT	CCoT-15	CCoT-30	CCoT-45	CCoT-60	CCoT-100
1 ⇒ 2	0.54	0.32	0.41	0.41	0.30	0.32
2 ⇒ 3	0.52	0.35	0.47	0.43	0.35	0.35
3 ⇒ 4	0.51	0.34	0.46	0.40	0.30	0.37
4 ⇒ 5	0.51	0.37	0.43	0.39	0.30	0.30
5 ⇒ 6	0.51	0.38	0.41	0.44	0.36	0.37
6 ⇒ 7	0.50	0.41	0.48	0.40	0.36	0.40
7 ⇒ 8	0.47	0.35	0.43	0.37	0.34	0.50

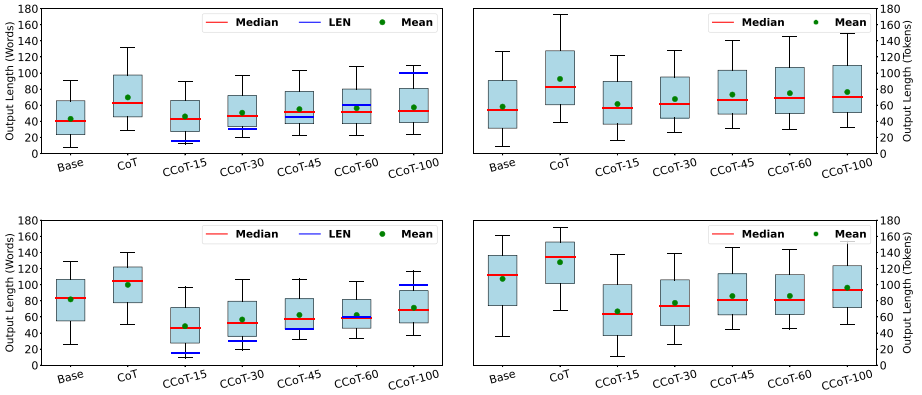


Fig. 6. Distribution of output lengths in terms of words (left plots) and tokens (right plots), between the 5th and 95th percentiles, for prompting strategies using the GSM8K test set. The top row shows Falcon-40B, and the bottom row shows LLaMA 2-70B.

the middle steps show the largest differences, especially for aggressive variations like CCoT-15 and CCoT-45, whereas early and late steps retain more information flow.

4.5. Ability to control the output length

The previous experiments examined how CCoT strategies can affect the accuracy and generation time on the average. However, despite the discussed benefits, it is also crucial to understand how CCoT prompting can effectively limit the output length for each addressed sample (**RQ3**). This can be useful for better tuning the length parameter in the CCoT prompt or identifying the conditions under which the proposed prompting strategy fails to compress the output. To evaluate the ability of an LLM to produce concise answers in response to a given prompt, we analyzed in Fig. 6 the output length (in terms of words and tokens) under different CCoT length constraints.

Fig. 6 shows the statistics of answer lengths, including both word lengths (left plots) and token lengths (right plots), produced by the evaluated models on the GSM8K test set. Each box plot represents the distribution of output lengths between the 5th and 95th percentiles across all tested samples. The blue line indicates the CCoT length constraint (shown only for the word-length plots, for consistency with the CCoT definition), the red line denotes the median, and the green dot represents the mean. Ideally, a model that respects the specified length constraint for each tested sample should have the entire distribution located below the blue line (left plots).

As depicted in Fig. 6, CoT-based LLMs tend to generate longer answers, along with longer token sequences if not explicitly constrained, which significantly impacts generation time. The imposed length constraint in the CCoT prompt has a notable effect on the output length, both in terms of generated words and tokens. However, in practice, LLMs are not always able to respect the given limit, especially for smaller values such as 15, 30, or 40, which prove to be more challenging.

To summarize, given the nature of the CCoT prompting, it is reasonable to consider a tolerance margin in respecting the requested length. To this end, in the following paragraphs we evaluate the considered models by the metrics proposed in Section 3.3, which extend the accuracy by also accounting for conciseness.

5. Discussion

From the experimental results, a key observation is that sufficiently large models, such as Falcon-40B and Llama2-70B, benefit the most from CCoT, achieving a better trade-off between accuracy and efficiency. However, we acknowledge that smaller models may struggle to maintain this trade-off, often producing incorrect answers when reasoning length is constrained. A detailed analysis of these findings is provided in Appendix B. We believe that this behavior aligns with the inherent capability of large language models to implicitly regulate the length of their generated responses [12].

Another important direction concerns the broader benefits of conciseness in LLMs beyond efficiency, which is the primary focus of this work. While conciseness is often justified in terms of reducing the number of reasoning steps (see Section 4.4), our analysis of redundancy and information flow suggests that conciseness can also emerge in scenarios where CCoT and CoT produce a similar number of reasoning steps.

These observations indicate that the proposed metrics could be further leveraged to explore additional benefits of concise reasoning, such as mitigating hallucinations or reducing error propagation. For instance, a lower Information Flow score may indicate that the model retains only the logically necessary information required to reach a correct answer step by step, while excluding superfluous details. This filtering process may help reduce the risk of error propagation by limiting the influence of unnecessary or irrelevant information during the reasoning process [31].

6. Conclusion

This work investigated the role of conciseness in the answers generated by LLMs for text-to-text reasoning tasks. To better evaluate this aspect, we introduced three metrics that jointly assess correctness and conciseness, enabling a more comprehensive evaluation of model performance under user-defined constraints.

To further study how prompting strategies affect the conciseness of model outputs, we proposed Constrained Chain-of-Thought, a prompt-engineering approach designed to provide greater control over the length of generated reasoning and answers. In addition, we explored the structural properties of concise reasoning by analyzing redundancy and information flow across reasoning steps. These analyses offer an initial perspective on how conciseness manifests within the embedding space of generated reasoning.

Overall, our findings highlight the importance of considering conciseness as a key dimension in the evaluation of LLM-generated reasoning. The proposed concise-accuracy metrics provide a principled way to assess the trade-off between correctness and output length, while CCoT offers a simple yet effective mechanism to encourage more concise reasoning behavior. These contributions may help guide future research toward developing LLM systems that are more efficient, predictable, and controllable.

As future work, we plan to extend this analysis to more complex reasoning settings, including open-domain question answering, commonsense reasoning, and multi-hop reasoning tasks. Moreover, to expand its generalizability, future research directions could explore automated methods for selecting metrics' user-defined parameters, enhancing the stability and adaptability of the model across diverse tasks.

CRediT authorship contribution statement

Sania Nayab: Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization. **Giulio Rossolini:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Formal analysis, Conceptualization. **Marco Simoni:** Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization. **Andrea Saracino:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Formal analysis, Conceptualization. **Giorgio Buttazzo:** Writing – review & editing, Supervision, Methodology, Formal analysis, Conceptualization. **Nicolamaria Manes:** Writing – review & editing, Methodology, Formal analysis, Conceptualization. **Fabrizio Giacomelli:** Writing – review & editing, Methodology, Formal analysis, Conceptualization.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Additional analysis of the conciseness

A.1. Analysis of word length and token length

Fig. A.7 shows that CoT prompting substantially affects the output-length behavior of Falcon-40b. The top plot reveals a strong linear relationship between tokens and words for both settings, indicating that CoT does not alter the basic correspondence between tokenized and lexical output length. However, the CoT variant exhibits a clear shift toward larger responses and a noticeably wider dispersion, particularly in the high-token regime. This suggests that CoT increases not only the average response length but also the variability of generation behavior across samples.

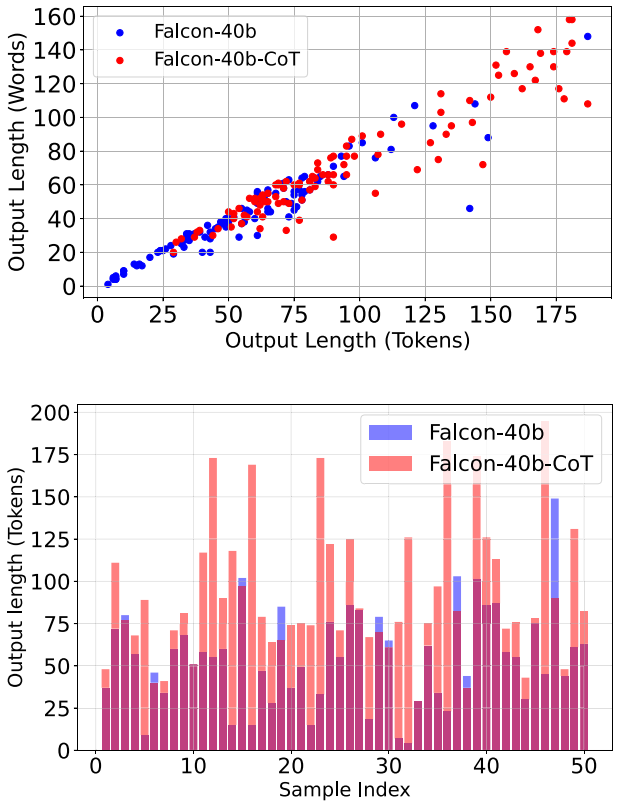


Fig. A.7. Analysis of the impact of CoT on Falcon-40b efficiency: (top) Relation between response time and output length (in terms of tokens), without CoT (blue dots) and with CoT (red dots), across 100 questions from the GSM8K test set. (bottom) Output tokens' variation between output length with CoT and without CoT using 50 random samples from the GSM8K test set.

The bottom plot (in Fig. A.7) further confirms that, for most GSM8K examples, CoT produces longer output tokens than the standard prompting setting. Overall, it indicates that CoT introduces a significant verbosity overhead, making reasoning more expensive in terms of token usage and less consistent in inference efficiency.

A.2. Analysis of the RMS

Figs. A.8, A.9, and A.10 present the RMS values for Llama2-70b and Falcon-40b across the three datasets analyzed in the paper. These results specifically highlight scenarios where the model successfully answers questions with the CCoT prompt but not with the CoT prompt. The purpose of these plots is to check whether the behavior of CCoT in terms of redundancy is coherent with that observed in Section 4.4. In addition, these plots help to investigate the potential correlation between the accuracy of the model's responses and the reduction in redundancy. The plots in the left column correspond to Llama2-70b, while those in the right column correspond to Falcon-40b. In each plot, we also highlight the interval between Q1 and Q3 for CoT and CCoT answers, along with the overlap range of these intervals.

As we can see from these plots, the redundancy behavior for CCoT and CoT closely resembles the trend observed in Section 4.4 calculated across all questions. This suggests that the reduction in redundancy (for Llama2-70b) and number of steps (for Falcon-40b) achieved by CCoT is consistent and provides more accurate and concise answers compared to CoT. In fact, the reduction in terms of redundancy or number of steps in the cases where CoT fails indicates that the CCoT prompt provides better guidance to the model, even in complex or ambiguous scenarios where CoT tends to struggle.

A.3. Information flow

Since the main paper examined the information flow in answers with 8 steps based on the CCoT answer distribution, we further demonstrate the effectiveness of CCoT by analyzing step-by-step information flow for another median step count, specifically 9 steps. Tables A.5 and A.6 present the scores for Llama2-70b on GSM8K and SVAMP, respectively.

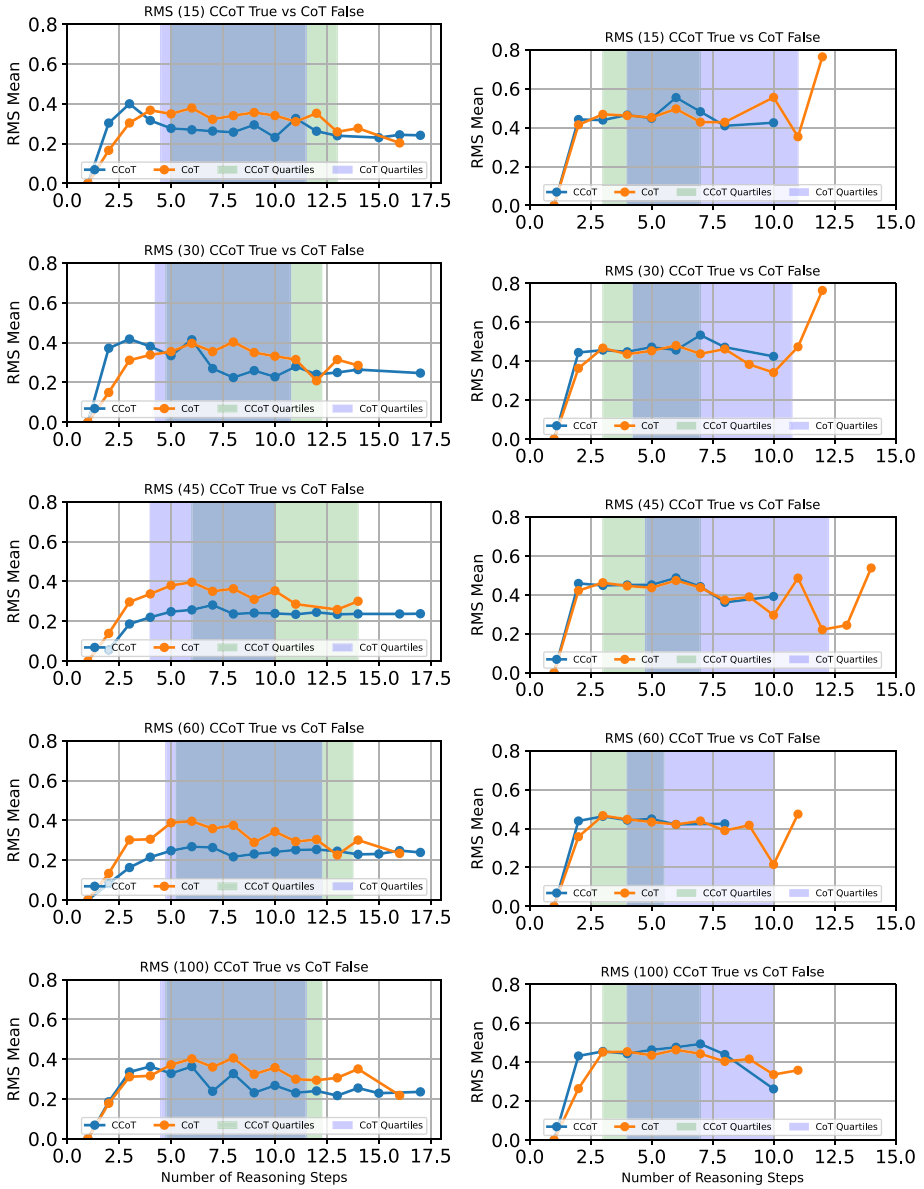


Fig. A.8. RMS mean score comparison between CCoT-true and CoT-false on the GSM8K dataset, with Llama2-70b (left side) and Falcon-40b (right side). The overlapped highlighted area illustrates the answer distribution between Q1 and Q3.

In Table A.5, for GSM8K, all steps show a reduced repetition of semantic information for CCoT, except for CCoT-45 at the initial step and CCoT-100 at the final step. A different behavior is observed at the final step for all CCoTs in the other dataset (SVAMP), as shown in Table A.6, although semantic information is retained throughout the steps.

We also calculated the semantic information flow for answers with 8 and 9 steps, based on the median step distribution of CCoT answers generated by Falcon-40b. The results are presented in Tables A.7 and A.8 for the GSM8K and SVAMP datasets, respectively. Interestingly, Falcon-40b exhibits contrasting behavior in terms of information scores, often displaying higher values, which suggests greater repetition of semantic information across steps. This behavior is likely influenced by its medium-scale architecture and the nature of its training dataset, which may not be well-suited for handling constrained reasoning tasks.

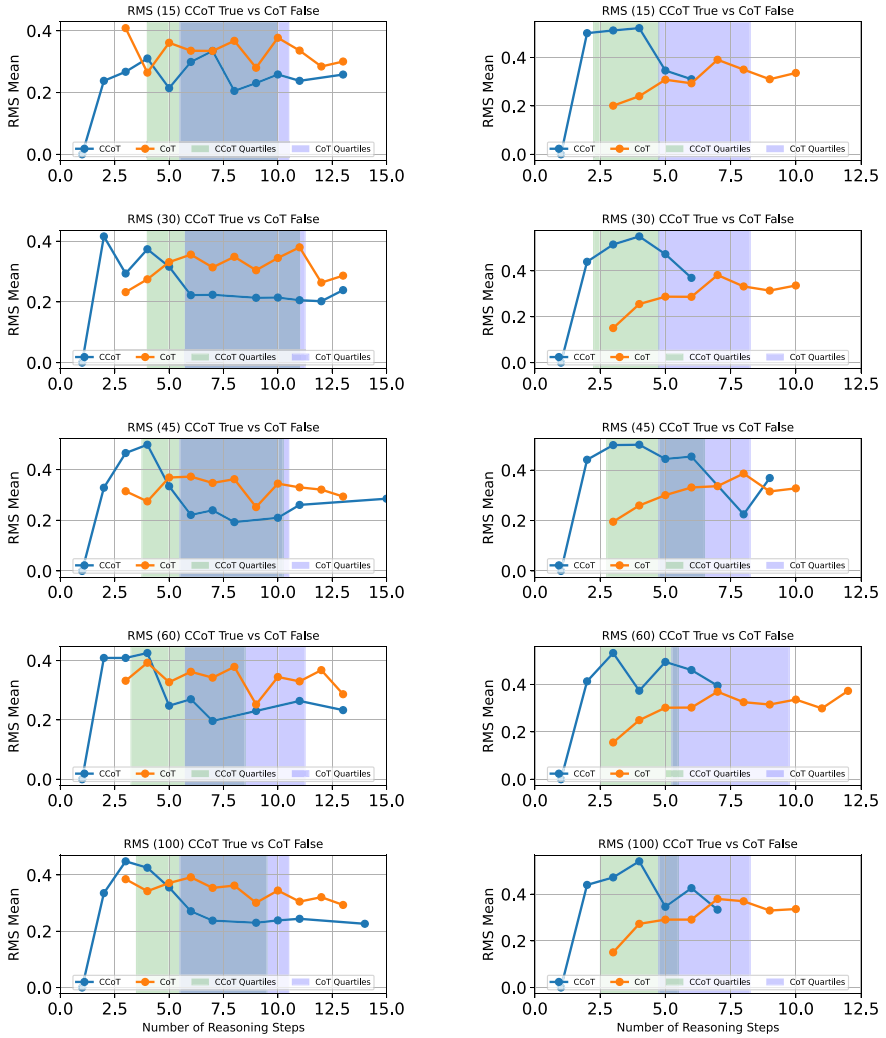


Fig. A.9. RMS mean score comparison between CCoT-true and CoT-false on the SVAMP dataset, with Llama2-70b (left side) and Falcon-40b (right side). The overlapped highlighted area illustrates the answer distribution between Q1 and Q3.

However, as also highlighted in the main text of the paper, it is important to note that Falcon-40b achieves improved conciseness due to the significantly lower average number of steps it produces. This allows for quicker reasoning decisions in many samples. This observation clarifies that the analysis reported in Tables A.7 and A.8 is inherently unbalanced, as the number of Falcon-40b answers with 8 and 9 steps under CCoT is smaller than those under CoT. Please note that in Table A.8, there is no information score for CCoT-15 because Falcon-40b does not generate answers with 8 or 9 steps when constrained to a reasoning length of up to 15 tokens.

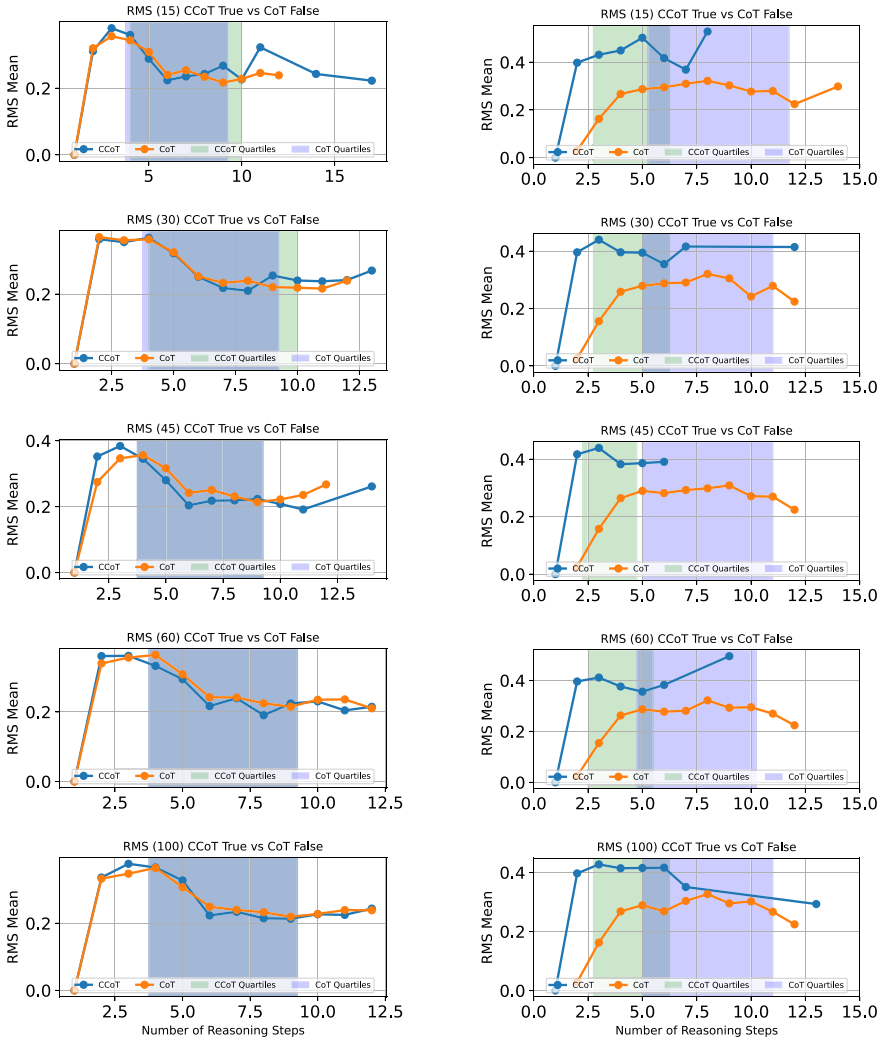


Fig. A.10. RMS mean score comparison between CCoT-true and CoT-false on the ASDIV dataset, with Llama2-70b shown (left side) and Falcon-40b (right side). The overlapped highlighted area illustrates the answer distribution between Q1 and Q3.

Table A.5
GSM8K Llama2-70b information flow mean values comparison across answers with 9 steps.

Steps	CoT	CCoT-15	CCoT-30	CCoT-45	CCoT-60	CCoT-100
1 ⇒ 2	0.48	0.29	0.32	0.48	0.45	0.33
2 ⇒ 3	0.54	0.38	0.40	0.35	0.33	0.36
3 ⇒ 4	0.49	0.36	0.37	0.40	0.38	0.35
4 ⇒ 5	0.49	0.35	0.36	0.38	0.37	0.34
5 ⇒ 6	0.51	0.38	0.40	0.39	0.38	0.36
6 ⇒ 7	0.49	0.37	0.37	0.39	0.36	0.36
7 ⇒ 8	0.50	0.36	0.37	0.43	0.40	0.35
8 ⇒ 9	0.55	0.50	0.54	0.45	0.51	0.61

Table A.6
SVAMP Llama70b information flow mean values comparison across answers with 9 steps.

Steps	CoT	CCoT-15	CCoT-30	CCoT-45	CCoT-60	CCoT-100
1 ⇒ 2	0.46	0.29	0.32	0.30	0.29	0.32
2 ⇒ 3	0.49	0.31	0.36	0.33	0.34	0.34
3 ⇒ 4	0.46	0.33	0.34	0.31	0.30	0.32
4 ⇒ 5	0.45	0.32	0.34	0.32	0.29	0.31
5 ⇒ 6	0.49	0.32	0.39	0.37	0.36	0.34
6 ⇒ 7	0.51	0.31	0.37	0.41	0.34	0.33
7 ⇒ 8	0.47	0.32	0.38	0.38	0.37	0.39
8 ⇒ 9	0.49	0.54	0.63	0.68	0.62	0.58

Table A.7
GSM8K Falcon-40b information flow mean values comparison.

(a) Answers with 8 steps

Steps	CoT	CCoT-15	CCoT-30	CCoT-45	CCoT-60	CCoT-100
1 ⇒ 2	0.58	0.65	0.64	0.65	0.61	0.59
2 ⇒ 3	0.62	0.69	0.68	0.70	0.68	0.64
3 ⇒ 4	0.60	0.74	0.76	0.74	0.68	0.63
4 ⇒ 5	0.59	0.66	0.63	0.64	0.64	0.66
5 ⇒ 6	0.61	0.62	0.72	0.67	0.66	0.64
6 ⇒ 7	0.60	0.61	0.66	0.68	0.68	0.62
7 ⇒ 8	0.59	0.50	0.55	0.54	0.56	0.59

(b) Answers with 8 steps

Steps	CoT	CCoT-15	CCoT-30	CCoT-45	CCoT-60	CCoT-100
1 ⇒ 2	0.55	0.62	0.69	0.61	0.58	0.67
2 ⇒ 3	0.58	0.62	0.65	0.65	0.57	0.73
3 ⇒ 4	0.54	0.62	0.67	0.63	0.61	0.73
4 ⇒ 5	0.54	0.62	0.64	0.46	0.55	0.63
5 ⇒ 6	0.54	0.69	0.74	0.67	0.64	0.76
6 ⇒ 7	0.55	0.70	0.59	0.48	0.56	0.66
7 ⇒ 8	0.55	0.65	0.77	0.68	0.65	0.75
8 ⇒ 9	0.59	0.52	0.60	0.46	0.48	0.68

Table A.8
SVAMP Falcon-40b Information Flow Mean Values Comparison.

(a) Answers with 8 Steps

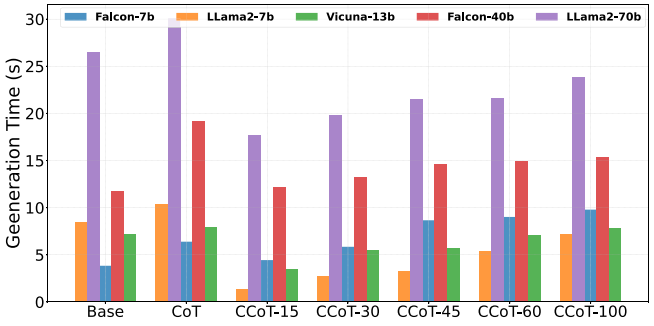
Steps	CoT	CCoT-15	CCoT-30	CCoT-45	CCoT-60	CCoT-100
1 ⇒ 2	0.26	0.60	0.58	0.55	0.58	0.61
2 ⇒ 3	0.68	0.58	0.53	0.53	0.60	0.52
3 ⇒ 4	0.60	0.68	0.52	0.60	0.49	0.54
4 ⇒ 5	0.61	0.56	0.51	0.56	0.63	0.60
5 ⇒ 6	0.63	0.51	0.65	0.55	0.68	0.64
6 ⇒ 7	0.63	0.95	0.76	0.52	0.64	0.62
7 ⇒ 8	0.55	0.73	0.57	0.60	0.56	0.65

(b) Answers with 9 Steps

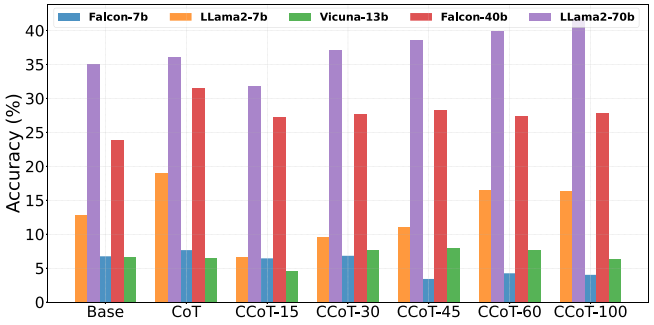
Steps	CoT	CCoT-15	CCoT-30	CCoT-45	CCoT-60	CCoT-100
1 ⇒ 2	0.26	–	0.64	0.62	0.56	0.61
2 ⇒ 3	0.65	–	0.59	0.55	0.53	0.66
3 ⇒ 4	0.57	–	0.66	0.52	0.62	0.63
4 ⇒ 5	0.58	–	0.74	0.67	0.55	0.54
5 ⇒ 6	0.60	–	0.67	0.47	0.67	0.61
6 ⇒ 7	0.57	–	0.86	0.47	0.75	0.64
7 ⇒ 8	0.63	–	0.77	0.69	0.63	0.62
8 ⇒ 9	0.54	–	0.61	0.56	0.57	0.43

Appendix B. Testing CCoT with smaller LLMs

In this experiment, we investigate the capability of larger set of LLMs to handle the CCoT prompting. Specifically, in Fig. B.11, we present an evaluation conducted on five LLMs, including small and medium-sized models such as Falcon-7b [32] and LLama2-7b [27]. The results acknowledge some difficulties in addressing the CCoT prompt when considering smaller models.



(a) Generation time



(b) Accuracy

Fig. B.11. Generation time (a) and accuracy (b) of five LLMs (Llama2-7b, Llama2-70b, Falcon-7b, Falcon-40b, and Vicuna-13b) on the GSM8K test dataset. Each model is evaluated using plain prompt (base), CoT, and CCoT with different length constraints.

We believe that such different outcomes of CCoT prompting can be attributed to various factors, such as the training data, the approach used to train the model, the model size, and the technique adopted during training. For instance, Llama2-70b is an autoregressive large-scale language model fine-tuned with human feedback, trained on a diverse combination of generic and open-source datasets. Such technical measures contribute to making CCoT effective in controlling the output length while improving the model's accuracy. The Falcon-40b model, in contrast, is smaller than Llama2-70b and trained on a different dataset (the dedicated RefinedWeb data [33]). While CCoT does not improve the accuracy of the model with respect to CoT, it still performs better than the base plain prompting, offering a trade-off by reducing generation times compared to CoT. Vicuna-13b also provides competitive results across different prompts, as it is a fine-tuned version of Llama2 and smaller than the previous Llama2-70b.

Conversely, small-scale LLMs, such as Falcon-7b and Llama2-7b, are not capable of properly handling the constrained prompting conditions in CCoT, resulting in higher generation times (as shown for Falcon-7b with large length values in CCoT) or incorrect answers with short CCoT values in Llama2-7b. This suggests that model size and training strategies severely impact the effectiveness of CCoT.

Considering the observations presented here, we believe that future directions could face potential training and fine-tuning strategies to integrate a better awareness and capability fo handling lengths also for smaller-sized models.

Data availability

I have used public datasets, and code is public. This is mentioned in the revised paper.

References

[1] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, T.B. Hashimoto, Stanford Alpaca: An Instruction-Following LLaMA Model, 2023.
[2] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J.E. Gonzalez, et al., Vicuna: an open-source chatbot impressing GPT-4 with 90%* ChatGPT quality, 2023, https://vicuna.lmsys.org (accessed 14 April 2023).
[3] F. Dolly, Introducing the World's First Truly Open Instruction-Tuned LLM. Databricks. Com, 2023.
[4] X. Geng, A. Gudibande, H. Liu, E. Wallace, P. Abbeel, S. Levine, D. Song, Koala: a dialogue model for academic research, Blog post, April 1 (2023).
[5] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q.V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, Adv. Neural Inf. Process. Syst. 35 (2022) 24824–24837.
[6] Y. Fu, L. Ou, M. Chen, Y. Wan, H. Peng, T. Khot, Chain-of-thought hub: a continuous effort to measure large language models' reasoning performance, arXiv preprint arXiv:2305.17306, 2023.
[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[8] S. Shekhar, T. Dubey, K. Mukherjee, A. Saxena, A. Tyagi, N. Kotla, Towards optimizing the costs of LLM usage, arXiv preprint arXiv:2402.01742, 2024.
[9] Z. Jiang, F.F. Xu, J. Araki, G. Neubig, How can we know what language models know? Trans. Assoc. Comput. Linguist. 8 (2020) 423–438.

[10] J. Kaplan, S. McCandlish, T. Henighan, T.B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei, Scaling laws for neural language models, arXiv preprint arXiv:2001.08361, 2020.
[11] K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, N.Z. Gong, Y. Zhang, et al., PromptBench: towards evaluating the robustness of large language models on adversarial prompts, arXiv preprint arXiv:2306.04528, 2023.
[12] A. Bhargava, C. Witkowski, M. Shah, M. Thomson, What's the magic word? A control theory of LLM prompting, arXiv preprint arXiv:2310.04444, 2023.
[13] S. Li, S. Zhao, Z. Zhang, Z. Fang, W. Chen, T. Wang, Basis is also explanation: interpretable legal judgment reasoning prompted by multi-source knowledge, Inf. Process. Manag. 62 (3) (2025) 103996.
[14] S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, et al., Language models (mostly) know what they know, arXiv preprint arXiv:2207.05221, 2022.
[15] H. Qiu, W. Mao, A. Patke, S. Cui, S. Jha, C. Wang, H. Franke, Z.T. Kalbarczyk, T. Başar, R.K. Iyer, Efficient interactive LLM serving with proxy model-based sequence length prediction, arXiv preprint arXiv:2404.08509, 2024.
[16] A. Azaria, T. Mitchell, The internal state of an LLM knows when its lying, arXiv preprint arXiv:2304.13734, 2023.
[17] Y. Liu, D. Li, K. Wang, Z. Xiong, F. Shi, J. Wang, B. Li, B. Hang, Are LLMs good at structured outputs? A benchmark for evaluating structured output capabilities in LLMs, Inf. Process. Manag. 61 (5) (2024) 103809.
[18] X. Wang, M. Salmani, P. Omidi, X. Ren, M. Rezagholizadeh, A. Eshaghi, Beyond the limits: a survey of techniques to extend the context length in large language models, arXiv preprint arXiv:2402.02244, 2024.
[19] C. Li, B. Bi, M. Yan, W. Wang, S. Huang, Addressing semantic drift in generative question answering with auxiliary extraction, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2021, pp. 942–947.
[20] Y. Liu, Z. Luo, K. Zhu, Controlling length in abstractive summarization using a convolutional neural network, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 4110–4119.
[21] S. Takase, N. Okazaki, Positional encoding to control output sequence length, arXiv preprint arXiv:1904.07418, 2019.
[22] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al., Training verifiers to solve math word problems, arXiv preprint arXiv:2110.14168, 2021.
[23] S. Bird, E. Klein, E. Loper, Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit, " O'Reilly Media, Inc.", 2009.
[24] J.W. Ratcliff, D.E. Metzener, et al., Pattern matching: the gestalt approach, Dr. Dobb's Journal 13 (7) (1988) 46.
[25] T. Zhang, V. Kishore, F. Wu, K.Q. Weinberger, Y. Artzi, BERTScore: evaluating text generation with BERT, CoRR arxiv:1904.09675, 2019.
[26] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805, 2018.
[27] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., LLaMA: open and efficient foundation language models, arXiv preprint arXiv:2302.13971, 2023.
[28] A. Patel, S. Bhattamishra, N. Goyal, Are NLP models really able to solve simple math word problems? arXiv preprint arXiv:2103.07191, 2021.
[29] S.-Y. Miao, C.-C. Liang, K.-Y. Su, A diverse corpus for evaluating and developing english math word problem solvers, arXiv preprint arXiv:2106.15772, 2021.
[30] Y. Fu, H. Peng, A. Sabharwal, P. Clark, T. Khot, Complexity-based prompting for multi-step reasoning, in: The Eleventh International Conference on Learning Representations, 2022.
[31] R. Li, Z. Luo, X. Du, Fine-grained hallucination detection and mitigation in language model mathematical reasoning, arXiv preprint arXiv:2410.06304, 2024.
[32] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, É. Goffinet, D. Hesslow, J. Launay, Q. Malartic, et al., The Falcon series of open language models, arXiv preprint arXiv:2311.16867, 2023.
[33] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, J. Launay, The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only, arXiv preprint arXiv:2306.01116, 2023.