CCQA: Generating Question from Solution Can Improve Inference-Time Reasoning in SLMs

Jin Young Kim[1] **Ji Won Yoon**[1†]
[1]Department of Artificial Intelligence, Chung-Ang University
{wlsdud338, jiwonyoon}@cau.ac.kr

Abstract

Recently, inference-time reasoning strategies have further improved the accuracy of large language models (LLMs), but their effectiveness on smaller models remains unclear. Based on the observation that conventional approaches often fail to improve performance in this context, we propose **C**ycle-**C**onsistency in **Q**uestion **A**nswering (CCQA), a novel reasoning method that can be effectively applied to SLMs. Inspired by cycle consistency, CCQA generates a question from each reasoning path and answer, evaluates each by its similarity to the original question, and then selects the candidate solution with the highest similarity score as the final response. Since conventional SLMs struggle to generate accurate questions from their own reasoning paths and answers, we employ a lightweight Flan-T5 model specialized for question generation to support this process efficiently. From the experimental results, it is verified that CCQA consistently outperforms existing state-of-the-art (SOTA) methods across eight models on mathematical and commonsense reasoning benchmarks. Furthermore, our method establishes a new practical baseline for efficient reasoning in SLMs. Source code can be found at https://github.com/scai-research/ccqa_official.

1 Introduction

Recent advancements in large language models (LLMs) have yielded remarkable performance across a wide range of tasks, including machine translation (Bahdanau et al., 2014; Stahlberg, 2020), code generation (Chen et al., 2021; Feng et al., 2020), sentiment analysis (Socher et al., 2013; Devlin et al., 2019), and reasoning (Shao et al., 2024; Bhargava and Ng, 2022). On top of that, inference-time reasoning strategies, such as chain-of-thought (CoT) (Wei et al., 2022), self-consistency (SC) (Wang et al., 2023), and self-correction (Huang et al., 2024), can produce more reliable outputs and further improve model accuracy, albeit at the cost of additional test-time computation (Wei et al., 2022; Wang et al., 2023; Huang et al., 2024).

While prior studies have clearly demonstrated the effectiveness of these reasoning techniques for large-scale models (Wang et al., 2023; Huang et al., 2024; Madaan et al., 2023), their applicability to small language models (SLMs) has yet to be fully explored. This motivates us to empirically investigate whether such reasoning strategies remain effective when applied to SLMs, and our observations indicate that they often lead to performance degradation in this setting, which will be discussed in Section 5.

The performance degradation observed in SLMs can be attributed to two main factors. First, smaller models could struggle to understand complex inputs and fail to follow instructions (Chang et al., 2024; An et al., 2024; Fang et al., 2024; Shi et al., 2024). However, recent self-feedback methods, such as self-correction (Huang et al., 2024), self-refinement (Madaan et al., 2023), and universal self-consistency (USC) (Chen et al., 2023), operate under the assumption that the model is capable of comprehending lengthy and complex inputs to generate appropriate feedback. This mismatch between the model's capacity and the underlying assumption often leads to suboptimal or even misleading outputs in the context of SLMs. Second, voting-based approaches such as SC (Wang et al., 2023) rely on a majority vote across multiple generated answers. This strategy becomes less effective when SLMs produce highly inconsistent outputs (Wang et al., 2024). In such cases where generated answers exhibit high variance without a clearly dominant response, majority voting fails to produce a reliable consensus and offers no meaningful advantage over random choice. This occurs because SC selects the final answer solely based on frequency, without

†Corresponding author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 13944–13956
November 4-9, 2025

evaluating the quality of reasoning paths.

To address these limitations, we propose a novel reasoning method that can be effectively applied to SLMs, called **C**ycle-**C**onsistency in **Q**uestion **A**nswering (CCQA). Inspired by the principle of cycle consistency (Hoffman et al., 2018), we construct a cycle between the original question, the solution produced by the SLM, and the question generated from that solution. Here, the solution includes a reasoning path and its corresponding answer. We believe that if the reasoning path and answer are correct, the regenerated question should be highly similar to the original input question. In the proposed framework, the SLM first receives the question as input and produces multiple candidate solutions. When there is no dominant response during majority voting, CCQA generates a new question from each candidate and measures its similarity to the original; a higher similarity score indicates that the solution is more likely correct. The candidate solution whose generated question most closely matches the original is selected as the final response, without requiring the model to process any additional complex input. Moreover, we fine-tune a lightweight Flan-T5-base (Chung et al., 2024) model to generate questions from candidate solutions. This is because conventional SLMs typically struggle to generate questions from their reasoning paths and answers. We confirm that our fine-tuned Flan-T5 is both efficient and excels at producing high-quality questions.

Extensive experiments are conducted on six reasoning benchmarks, including four mathematical and four commonsense tasks. Our evaluation uses eight SLMs ranging from 135M to 3B parameters, including Llama3.2 (Grattafiori et al., 2024), SmolLM2 (Allal et al., 2025), and Qwen2.5 (Yang et al., 2025). From the experimental results, it is confirmed that CCQA consistently outperforms current state-of-the-art (SOTA) reasoning methods across most SLMs and benchmarks. Notably, CCQA with Llama3.2-3B on GSM8K achieves 69.60% accuracy compared to USC's 53.83%. On CommonSenseQA with Llama-1B, it attains 38.74% versus USC's 33.99%, demonstrating its effectiveness in enhancing SLM reasoning capabilities.

Our main contributions are summarized as follows:

- Our paper introduces a novel inference-time reasoning technique for SLMs, namely CCQA, that evaluates the quality of each reasoning path and its answer by regenerating a question and measuring its similarity to the original. *To the best of our knowledge, this is the first attempt to investigate the inference-time reasoning capabilities of SLMs and to improve them.*
- We leverage a lightweight Flan-T5 model to generate questions from candidate solutions. Compared to conventional SLMs, our fine-tuned Flan-T5 is computationally efficient and produces higher-quality questions.
- Our extensive experiments across diverse benchmarks and SLMs demonstrate that CCQA consistently outperforms SOTA reasoning methods, substantially improving reasoning capabilities of SLMs.

2 Related Work

2.1 Reasoning Methods for LLMs in Test-time

Reasoning remains one of the most challenging tasks for language models, involving complex problem-solving such as arithmetic and commonsense reasoning. Various approaches have been developed to enhance reasoning performance. CoT (Wei et al., 2022) induces models to describe problem-solving steps clearly, with extensions like least-to-most (Zhou et al., 2023) and tree of thought (Yao et al., 2023) exploring more diverse reasoning paths. Self-feedback methods, including self-correction (Huang et al., 2024) and self-refinement (Madaan et al., 2023), enable models to improve outputs using their own responses, though these typically require processing extensive input contexts. Aggregation techniques such as SC (Wang et al., 2023) employ majority voting across multiple samples, while USC (Chen et al., 2023) generates responses by considering all previous outputs. However, these methods rely on assumptions about model capacity that become problematic for SLMs with inconsistent outputs or limited ability to process complex prompts.

2.2 Cycle Consistency in Generative Models

Cycle consistency has been widely used as an effective training and evaluation paradigm across various domains. Initially introduced in computer vision for tasks such as image-to-image translation (Zhu et al., 2017) and 3D reconstruction (Tulsiani

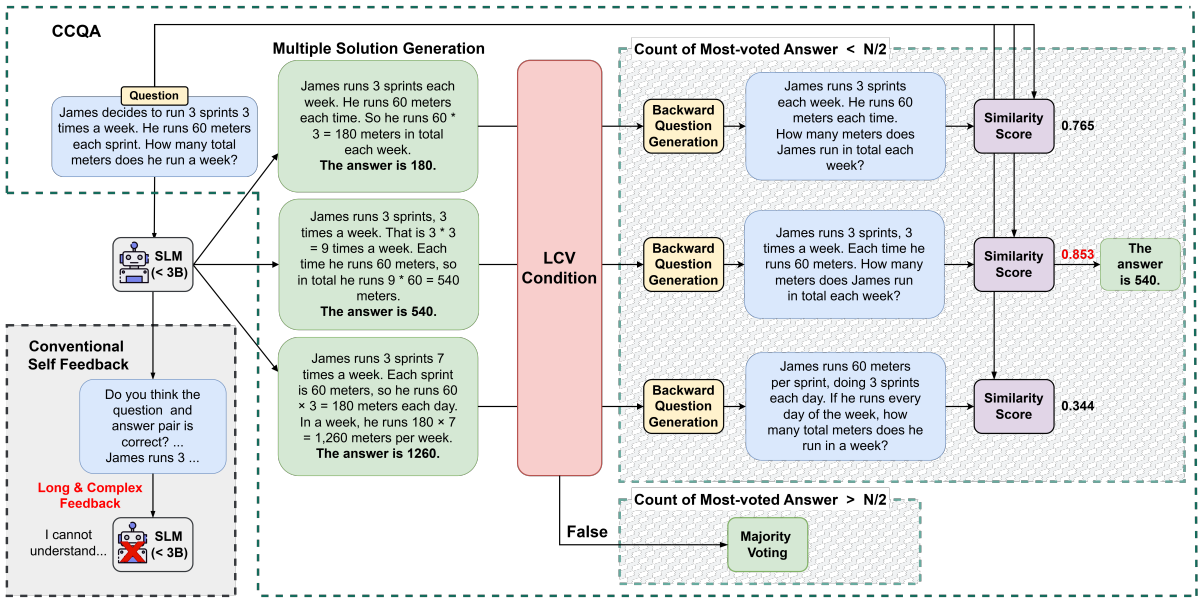

Figure 1: Overall process of the CCQA. (1) CCQA receives a question as input and generates N solutions. (2) It checks for the LCV condition; if the LCV condition is met (i.e., when the model's answers are inconsistent with no clear majority), it regenerates questions from the answers, otherwise it performs majority voting to select the final answer. (3) Under the LCV condition, it compares the generated questions with the original question to assign similarity scores. (4) The solution corresponding to the question with the highest similarity score is selected as the final answer.

et al., 2018), the concept leverages the principle that transformations should be reversible — if data is transformed from domain A to domain B and back to domain A, the result should closely match the original input. This principle has been extended to natural language processing, including machine translation (Sennrich et al., 2016; He et al., 2016), where back translation serves as a form of cycle consistency to improve translation quality. Recent work has also explored cycle consistency for evaluating text generation quality (Lee and Lee, 2022) and ensuring factual consistency in summarization (Yuan et al., 2020). While cycle consistency has been used to assess generation quality in various domains, applying it to SLM's reasoning quality offers a promising new direction.

3 Proposed Method

3.1 Motivation

SLMs have gained increasing attention (Qu et al., 2025; Liu et al., 2024b), but when applied to reasoning tasks, they underperform mainly due to two limitations. First, SLMs struggle with processing long and complex inputs (>1K tokens) due to their weak in-context learning abilities (Liu et al., 2024a; An et al., 2024; Fang et al., 2024). Second, they often generate inconsistent outputs that are highly varied (Wang et al., 2024). Consequently, both self-feedback mechanisms (e.g., self-correction, self-refinement) and voting mechanisms (e.g., SC, USC) show limited effectiveness with SLMs, as they require either strong input processing capabilities or output consistency. Despite these limitations, reasoning approaches specifically designed for SLMs remain largely unexplored, highlighting the need for tailored methodologies for SLMs. Based on these observations, we derive two key requirements for effective SLM reasoning: (1) avoid lengthy feedback or correction prompts, and (2) reliably identify high-quality reasoning despite inconsistent outputs.

3.2 CCQA

The overall process of CCQA is illustrated in Figure 1. CCQA begins by generating N independent solutions, including both reasoning paths (RPs*) and answers, using CoT prompting. It then applies answer-only voting, as in the SC method, disregarding RPs. However, SLMs frequently produce extremely diverse answers to the same question, leading to unstable voting patterns where majority voting acts like random selection. In those cases, we employ a fine-tuned T5 model to regenerate

*Reasoning paths refer to the step-by-step solution processes generated using CoT prompting, excluding the final answer.

a question from each solution (Section 3.4) and measure both lexical and semantic similarity between each generated question and the original one. Finally, CCQA determines the final output by selecting the answer whose generated question has the highest similarity score (Section 3.5).

3.3 Multiple Solution Generation and LCV Identification

When SLMs generate highly varied answers, additional verification becomes necessary. For instance, consider a case where a mathematical problem yields answers '18', '24', '27', and '35' with similar frequencies across multiple generated solutions. In this scenario, it is difficult to determine which answer is more reliable based on voting alone, as the method cannot evaluate the quality of the RPs. To address this problem, we define such situations with highly varied answers as **L**ow **C**onfidence **V**oting (LCV) conditions. An LCV condition is defined by the following condition:

$$\text{LCV} = \left\{ \max_j \text{freq}(A_j) < \lceil N/2 \rceil \right\}.$$

Here, $\text{freq}(A_j)$ represents the frequency of the j-th unique answer, and N is the total number of generated responses. In other words, LCV is defined as a situation where the frequency of the most voted answer does not reach a majority of the total number of responses. In our experiments on the GSM8K dataset, evaluating eight different models of various sizes (0.5B-3B), we found that on average, LCV occurred in 36.46% of problems, and 80.85% of answers selected by SC in these LCV cases were incorrect. This demonstrates that a simple majority voting method cannot sufficiently leverage the reasoning capabilities of SLMs. In these situations, a verification mechanism that directly considers RPs is needed rather than a majority voting approach. Therefore, we apply backward question generation and similarity measurement methods to directly evaluate the quality of each RP.

3.4 Backward Question Generation

To evaluate RP quality in LCV situations, we leverage backward question generation. The backward question generation process is as follows: In LCV situations, each RP_i (where i indicates the index of the RP) is used as input to the fine-tuned T5 model to generate a question (GQ_i, where i indicates the index of the GQ). To ensure more accurate question generation, we carefully select the appropriate

Algorithm 1 CCQA

Require: Original Question OQ, parameter α, β, sample count N, backward question generation BQG, reasoning path RP_i, answer A_i
Ensure: Final answer (RP_{final}, A_{final})

```
Generate reasoning path and answers {(RP_i, A_i)}_{i=1}^N
Count frequency of each unique answer: freq(A_j)
j_max ← arg max_j freq(A_j)
if freq(A_{j_max}) ≥ ⌈N/2⌉ then
    return (RP_{j_max}, A_{j_max})
else
    for i = 1 to N do do
        GQ_i ← BQG(RP_i)
        bleu_i ← BLEU(GQ_i, OQ)
        cos_i ← Cosine similarity(GQ_i, OQ)
        score_i ← α · bleu_i + β · cos_i
    end for
    best_idx ← arg max_i score_i
    return (RP_{best_idx}, A_{best_idx})
end if
```

model architecture and design a comprehensive training process. Among models of similar size, we choose the T5-base model due to its superior performance in text generation tasks. We also experiment with other small-sized models, but they do not perform well regardless of whether we apply fine-tuning or not. Furthermore, we utilize training sets from various mathematical and commonsense reasoning benchmarks, reverse the existing question-answer pairs to answer-question format for our task. Detailed hyperparameters and data preprocessing rules are presented in the Section 4.

3.5 Similarity-based Answer Selection

After the backward question generation is completed, we need to compare the similarity between the generated questions (GQ) and the original question (OQ). We measured the similarity between each GQ_i from each reasoning path RP_i and the original question (OQ) using two complementary methods: BLEU (Papineni et al., 2002) and embedding-based cosine similarity (Reimers and Gurevych, 2019). BLEU score for lexical overlap and embedding-based cosine similarity for semantic correspondence. BLEU score captures the lexical overlap and structural similarity by measuring n-gram matches between the generated and original questions. This helps identify how well the

surface-level textual elements are preserved. For semantic similarity, we used cosine similarity of sentence embeddings generated by Sentence-BERT (Reimers and Gurevych, 2019), which captures the overall meaning correspondence between the two questions beyond exact word matches. These two measurements were combined using the following weighted sum:

$$\text{score}(GQ_i, OQ) = \alpha \cdot \text{BLEU}(GQ_i, OQ) + \beta \cdot \text{cosine}(GQ_i, OQ). \quad (1)$$

Here, BLEU is the BLEU score value, and cosine is the embedding-based cosine similarity score value. α and β are weights that adjust the importance of each measurement. In our method, we set α to 0.4 and β to 0.6. Detailed experiments for determining these weights are presented in Section 6. The complete CCQA approach is formalized in Algorithm 1.

4 Experimental Setup

Models. The specific models used in our experiments are Llama3.2-1B and Llama3.2-3B (Grattafiori et al., 2024), Qwen2.5-0.5B, Qwen2.5-1.5B, and Qwen2.5-3B (Yang et al., 2025), SmolLM2-135M and SmolLM2-360M (Allal et al., 2025). Llama3.2 is a decoder-only language model with improved reasoning and instruction-following capabilities. We selected Llama3.2-1B and Llama3.2-3B variants to test performance on recent architectural designs. Qwen2.5 is a transformer-based model known for its strong multilingual capabilities and performance on knowledge-intensive tasks. To assess how CCQA's scaling properties are affected by increasing model capacity, we utilized three different variants of this model. SmolLM2 is a lightweight model optimized for efficiency with a specialized architecture for resource-constrained environments. We included SmolLM2-135M and SmolLM2-360M variants to test CCQA's applicability in on-device environments. Also we fine-tuned Flan-T5-base(258M) models to generate question, using learning rate of 2e-5, 3 epochs, and a batch size of 16.

Benchmarks. We evaluated CCQA on six standard reasoning benchmarks. For arithmetic reasoning, we utilized GSM8K (Cobbe et al., 2021) with its multi-step grade school math problems (train: 7.47K, test: 1.32K), SVAMP (Patel et al., 2021) offering varied math word problems (train: 700, test: 300), and Multi-Arith (Roy and Roth, 2015) for problems requiring multiple operations (train: 420, test: 180). For commonsense reasoning, we selected CSQA (Talmor et al., 2019) for its multiple-choice questions requiring world knowledge (train: 9.74K, val: 1.22K, test: 1.14K), StrategyQA (Geva et al., 2021) which poses yes/no questions needing strategic inference (train: 1.6K, test: 687), and ARC-Challenge (Clark et al., 2018) (train: 1.12K, val: 299, test: 1.17K). If there was an answer field, we used the test dataset; if not, we used the dev dataset. Also, the datasets used for finetuning were the train sets of CSQA, StrategyQA, and GSM8K-main.

Implementation. We conducted experiments using the A6000 with 48GB. For generating model responses, we followed standard guidelines to set the temperature parameter for text generation (Radford et al., 2019; Holtzman et al., 2020). Specifically, we configured the temperature to 0.7 across all models when generate solutions. Additionally, based on previous research showing that top-p sampling provides more stable results for smaller models(Albalak et al., 2024; Brown et al., 2020), we used top-p = 0.9 for decoding. (Albalak et al., 2024; Brown et al., 2020). We conducted all experiments in a few-shot setting, utilizing demonstration examples derived from prior open-domain text generation studies (Wei et al., 2022; Wang et al., 2023). We also created simple prompts for T5 question generation. Our prompts and sample solutions are presented in Appendix A.1, which shows the corresponding prompts used for question generation. Additionally, we converted question-answer pairs from the training sets of various reasoning benchmarks into answer-question pairs to fine-tune the Flan-T5 model.

5 Experimental Results

We evaluated CCQA on multiple benchmarks, comparing it with conventional reasoning methods, including CoT (Wei et al., 2022), self-correction (Huang et al., 2024), SC (Huang et al., 2024), and USC (Chen et al., 2023). As mentioned earlier, we focused on assessing the effectiveness of inference-time reasoning strategies for SLMs.

5.1 Main Results

Arithmetic Reasoning. The results for arithmetic reasoning are presented in Table 1. Each SLM independently generated five solutions

Model	GSM8K					MultiArith					SVAMP				
	Base	CoT	Self-Corr	SC	CCQA	Base	CoT	Self-Corr	SC	CCQA	Base	CoT	Self-Corr	SC	CCQA
Qwen-0.5B	2.65	11.45	4.55	**17.32**	**17.32**	8.89	40.00	11.24	51.11	**52.22**	6.33	44.33	15.33	52.67	**55.00**
Qwen-1.5B	8.19	37.98	22.37	44.88	**48.37**	21.67	95.00	66.85	**97.22**	**97.22**	3.00	74.33	38.00	83.67	**84.00**
Qwen-3B	9.78	33.01	0.30	29.12	**30.71**	42.22	75.56	0.00	81.11	**82.78**	3.33	86.00	16.67	88.00	**88.33**
Llama-1B	2.05	25.32	17.89	35.78	**39.20**	8.89	70.22	25.84	85.00	**86.11**	2.66	52.33	40.00	58.67	**59.00**
Llama3.2-3B	1.59	49.81	4.85	69.31	**69.60**	21.11	80.58	8.63	93.89	**98.89**	4.66	79.00	27.33	85.00	**86.00**
Falcon-1B	5.76	32.52	0.08	40.94	**42.61**	7.22	79.21	63.33	91.67	**92.78**	3.33	44.00	20.67	51.33	**52.33**
SmolLM2-135M	1.90	2.35	0.00	1.97	**2.88**	0.56	0.00	0.00	**3.33**	**3.33**	5.00	6.00	0.00	**8.33**	7.67
SmolLM2-360M	2.65	6.60	0.00	**8.79**	8.72	1.67	7.78	0.00	24.44	**25.56**	1.66	15.33	0.00	24.33	**27.00**

Model	CommonSenseQA					StrategyQA					ARC-Challenge				
	Base	CoT	Self-Corr	SC	CCQA	Base	CoT	Self-Corr	SC	CCQA	Base	CoT	Self-Corr	SC	CCQA
Qwen-0.5B	40.33	39.64	21.21	43.00	**43.82**	52.33	53.13	19.07	**54.29**	**54.29**	44.96	44.28	26.54	48.46	**49.89**
Qwen-1.5B	57.56	62.74	7.53	**66.58**	66.34	51.38	55.02	12.23	52.55	**55.17**	66.19	71.33	24.66	**75.09**	74.40
Qwen-3B	65.26	65.11	70.27	**70.52**	**70.52**	54.15	51.97	51.53	52.98	**55.31**	76.19	79.95	26.56	84.90	**84.98**
Llama-1B	24.07	30.71	22.43	37.92	**38.74**	53.28	54.00	3.64	57.21	**57.35**	45.34	44.71	39.25	49.40	**49.66**
Llama3.2-3B	43.39	56.18	48.37	65.68	**66.42**	48.47	45.65	**51.53**	49.20	49.20	68.63	69.71	72.05	**74.40**	74.06
Falcon-1B	28.32	32.68	0.00	35.14	**35.79**	54.04	58.52	6.11	58.52	**59.57**	51.96	54.52	30.72	55.38	**55.72**
SmolLM2-135M	16.09	16.79	0.00	17.69	**18.35**	48.47	49.05	0.87	**49.34**	**49.34**	17.41	22.53	18.00	23.12	**23.98**
SmolLM2-360M	18.96	19.66	0.00	19.49	**19.82**	39.74	49.49	0.15	49.20	**49.49**	16.09	16.81	**18.21**	18.00	18.09

Table 1: Performance comparison including baseline and various inference-time techniques on arithmetic(GSM8K, Multi-Arith, SVAMP) and common-sense(CommonSenseQA, StrategyQA, ARC-Challenge) benchmarks, measured by accuracy(%). Base: baseline performance using greedy decoding, CoT: chain-of-thought prompting, Self-Corr: self-correction, SC: self-consistency, CCQA: proposed method.

($N = 5$) per question. Interestingly, conventional feedback-based methods, such as self-correction and USC, showed significant performance degradation. When applied to SLMs, these methods achieved markedly lower accuracy than chain-of-thought prompting and self-consistency, suggesting that feedback-dependent inference strategies may be suboptimal in the smaller model setting. However, the results show that CCQA consistently achieves the highest accuracy across most configurations, outperforming all other methods on GSM8K, SVAMP, and MultiArith. In particular, on the MultiArith benchmark, Llama3.2-3B with CCQA achieves a 5.00 % performance improvement over SC, showed better accuracy than the other methods. For SmolLM2, its limited capacity resulted in generally poor performance on the arithmetic reasoning benchmark. Nevertheless, CCQA still produced a measurable accuracy improvement.

Commonsense Reasoning. Experimental results for commonsense reasoning are also summarized in Table 1. Across the three evaluated benchmarks, including CommonsenseQA, StrategyQA, and ARC-Challenge, CCQA outperformed competing methods across most models and benchmarks. As with arithmetic reasoning, self-correction and USC also exhibited performance degradation in commonsense reasoning. For SmolLM2, every method except CCQA and SC performed worse than CoT, with some approaches' accuracy even falling to 0 %. Among the techniques, only CCQA and SC maintained performance at or above CoT. However, CCQA consistently delivered higher accuracy than SC. This suggests that CCQA provides more consistent gains for SLMs across both arithmetic and commonsense reasoning tasks, whereas other inference-time methods may sometimes underperform or show unstable results.

Benchmark	LCV	SC_{LCV}	$CCQA_{LCV}$	Δ
GSM8K	36.46	19.15	**22.11**	**+2.96**
CSQA	21.28	26.48	**28.68**	**+2.20**
StrategyQA	5.79	36.88	**48.03**	**+11.15**
SVAMP	36.46	19.15	**21.20**	**+2.05**

Table 2: Proportion of questions triggering the LCV condition and corresponding accuracy. LCV (%) = percentage of all samples under LCV condition (i.e., no clear majority); SC_{LCV} (%) = accuracy of self-consistency on LCV samples; $CCQA_{LCV}$ (%) = accuracy of the proposed method on the same samples; Δ (percentage-point gain) = $CCQA_{LCV} - SC_{LCV}$.

5.2 CCQA Performs More Robustly Under LCV Condition

We also compared SC and CCQA performance under the LCV condition, as reported in Table 2. When the LCV condition was true, this means that the model produced highly diverse and inconsistent answers without any clear majority. On StrategyQA benchmark, CCQA correctly solved 11.15% more problems under the LCV condition compared to SC. Because SC lacked a mechanism for resolving con-

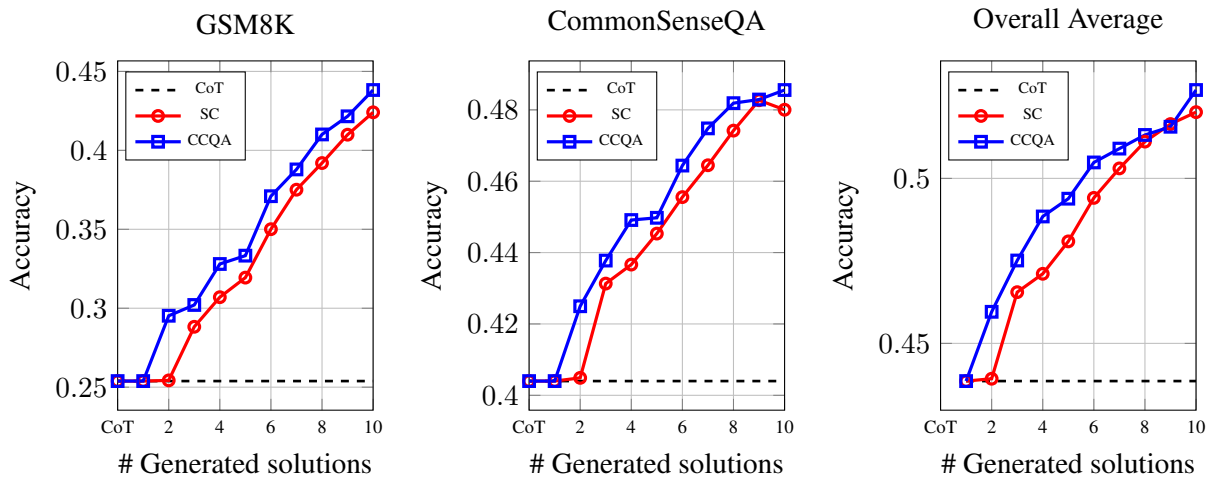

Figure 2: Comparison of CoT, SC, and CCQA accuracy across benchmarks for varying numbers of generated solutions N. The rightmost plot shows the mean performance across all benchmarks and models. Self-correction and USC, which performed worse than SC and CCQA (see Table 1), were omitted for clarity.

flicting answers, it struggled when outputs are inconsistent. In contrast, CCQA effectively selected higher-quality reasoning paths, demonstrating its potential as an inference-time strategy for SLMs. Though CCQA required slightly more computational resources than SC, it achieved a favorable performance-resource balance, offering significant accuracy gains with only marginal additional computational cost.

5.3 Robust Performance across Various Numbers of Responses

We evaluated CCQA's robustness by progressively increasing the number of generated solutions to 10 and measuring performance at each increment. As shown in Figure 2, we observed consistent performance gains across both arithmetic reasoning and commonsense benchmarks relative to SC, which was the strongest-performing method among all approaches aside from CCQA. The rightmost graph compared CCQA, CoT, and SC using the average of the six benchmarks we used. From the results, it is verified that CCQA demonstrates consistent performance improvements across all benchmark averages.

6 Analysis

6.1 Similarity Metrics for CCQA

To measure the similarity between generated questions and original questions, we considered various similarity metrics. First, we believed that using both surface-level and semantic similarities would be beneficial for the similarity score. This approach

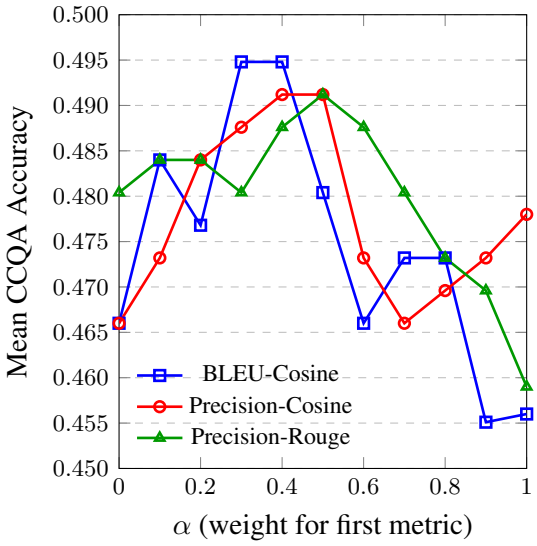

Figure 3: Comparison of different similarity metrics with varying weights (α for first metric, $\beta = 1 - \alpha$ for second metric).

provides a more comprehensive evaluation framework by capturing different aspects of textual similarity. Surface-level metrics can effectively identify exact matches and structural similarities, while semantic measures can recognize paraphrases and conceptually equivalent expressions that might use different vocabulary. Therefore, we employed BLEU and Rouge (Lin, 2004) for surface-level similarity, while utilizing embedding-based cosine similarity, BERTScore (Zhang et al., 2020) for semantic similarity. We found optimal performance by using a weighted sum of these surface-level and semantic similarity measures.

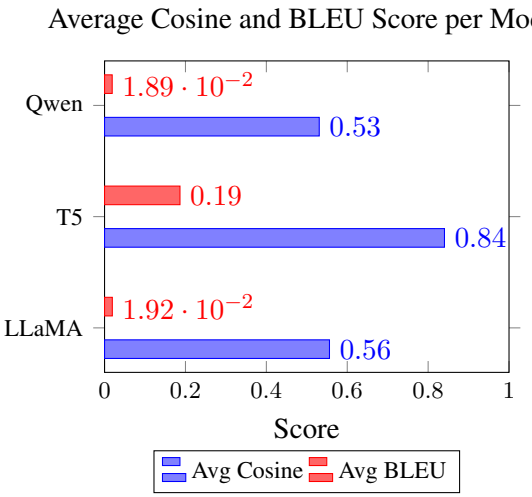

Figure 4: Average Cosine and BLEU Scores across LLaMA, T5, and Qwen models.

Our empirical analysis revealed that a balanced combination of lexical structure and semantic meaning provides the most effective similarity measure for identifying accurate reasoning paths. Specifically, assigning weights of $\alpha = 0.4$ to bleu score, and $\beta = 0.6$ to embedding-based cosine similarity yielded optimal performance across diverse reasoning benchmarks. These weights were determined through comprehensive grid search over range 0.0, 0.1,...,1.0 with the constraint $\alpha + \beta = 1$. The three combinations with the best performance are presented in Figure 3. Using this optimized similarity measure, CCQA selects the reasoning path and its corresponding answer that generates the question most similar to the original problem.

6.2 Backward Question Generation Model

For the efficiency and performance of the proposed method, generating backward questions played an important role. We considered several models requiring minimal additional resources, taking into account the characteristics of SLMs. We used a total of three models: Llama3.2-1B, Qwen2.5-0.5B, and Flan-T5. We first used these three models without fine-tuning, but all models failed to generate problems properly. Specifically, Llama3.2-1B and Qwen2.5-0.5B often generated responses that were irrelevant to the answers, while the T5 model generated questions but sometimes missed important parts of the answers. Therefore, we used the Flan-T5 model with fine-tuning. We also tried using other models with fine-tuning, but they exhibited the same problems. Detailed examples of question generation from all three models are presented in Appendix A.2. Additionally, we measured the similarity between the questions generated by the models and the original questions, not just through observation. As shown in Figure 2 below, when we generated questions from solutions by using Flan-T5 models, it had the highest average semantic similarity value. As a result, by using this model, we were able to improve performance with CCQA while only slightly increasing resource requirements.

7 Conclusion

We presented CCQA, a novel inference-time reasoning framework designed for SLMs. Inspired by the cycle consistency, CCQA regenerated a question from each candidate solution using a lightweight, fine-tuned Flan-T5 and compared it to the original prompt to identify the most reliable reasoning path. This simple yet effective mechanism makes the proposed method robust under LCV conditions, where small models typically produce inconsistent outputs, while adding only minimal computational overhead. From extensive experiments across arithmetic and commonsense benchmarks, it is verified that that CCQA consistently surpassed existing inference-time strategies, substantially enhancing the reasoning capabilities of SLMs.

Limitations

Despite its strong performance, CCQA has several limitations. First, the effectiveness of the proposed framework depends on the quality of the backward question generator; if the component produces low-quality questions, then CCQA's overall performance degrades. Second, the auxiliary Flan-T5 model introduces additional parameters. However, its lightweight design and the substantial performance gains on SLMs make this overhead acceptable. Also, considering that SLMs typically struggle to generate reliable questions on their own, the additional cost is essential for achieving robust reasoning performance. Compared to the high computational cost and numerous forward passes of other inference-time reasoning methods, CCQA's extra demand is reasonable. Finally, our evaluation is limited to arithmetic and commonsense reasoning in English, leaving broader domains for future work. Despite these limitations, we believe CCQA can substantially enhance the reasoning capabilities of SLMs and, by extension, improve their real-world utility.

Ethics Statement

This work does not raise any ethical concerns.

Acknowledgement

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2025-00515722). This work was also supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [RS-2021-II211341, Artificial Intelligent Graduate School Program (Chung-Ang University)].

References

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. 2024. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, and 3 others. 2025. Smollm2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*.

Chenxin An, Jun Zhang, Ming Zhong, Lei Li, Shansan Gong, Yao Luo, Jingjing Xu, and Lingpeng Kong. 2024. Why does the effective context length of llms fall short? *arXiv preprint arXiv:2410.18745*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Prajjwal Bhargava and Vincent Ng. 2022. Commonsense knowledge reasoning and generation with pre-trained language models: A survey. In *Proc. AAAI*, volume 36.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Ernie Chang, Matteo Paltenghi, Yang Li, Pin-Jie Lin, Changsheng Zhao, Patrick Huber, Zechun Liu, Rastislav Rabatin, Yangyang Shi, and Vikas Chandra. 2024. Scaling parameter-constrained language models with quality data. In *Proc. EMNLP*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tai, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(1).

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun (Jane) Qi, Scott Nickleach, Diego Socolinsky, "SHS" Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large language models (llms) on tabular data: Prediction, generation, and understanding — a survey. *Transactions on Machine Learning Research*.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and 1 others. 2020. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Proc. NeurIPS*, volume 29.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. 2018. Cycada: Cycle-consistent adversarial domain adaptation. In *Proc. ICML*.

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

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *Proc. ICLR*.

Seungyeon Lee and Minho Lee. 2022. Type-dependent prompt cycleqag: Cycle consistency for multi-hop question generation. In *Proc. COLING*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL Workshop*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, and 1 others. 2024b. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *Proc. ICML*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Proc. NeurIPS*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proc. NAACL-HLT*.

Guanqiao Qu, Qiyuan Chen, Wei Wei, Zheng Lin, Xianhao Chen, and Kaibin Huang. 2025. Mobile edge intelligence for large language models: A contemporary survey. *IEEE Communications Surveys & Tutorials*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8).

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proc. EMNLP-IJCNLP*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proc. EMNLP*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proc. ACL*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Zhenmei Shi, Junyi Wei, Zhuoyan Xu, and Yingyu Liang. 2024. Why larger language models do in-context learning differently? In *Proc. ICML*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP*.

Felix Stahlberg. 2020. Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proc. NAACL-HLT*.

Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. 2018. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *Proc. CVPR*.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large language models are not fair evaluators. In *Proc. ACL*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *Proc. ICLR*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proc. NeurIPS*, volume 35.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025. Qwen2.5 technical report. *arXiv preprint arXiv:2501.18189*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Proc. NeurIPS*, volume 36.

Li Yuan, Francis Eng Hock Tay, Ping Li, and Jiashi Feng. 2020. Unsupervised video summarization with cycle-consistent adversarial lstm networks. *IEEE Transactions on Multimedia*, 22(10).

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *Proc. ICLR*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *Proc. ICLR*.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. ICCV*.

A Appendix

A.1 Detailed Prompts And Inputs of Question Models

Table 3 gives simple prompt examples to generate questions. We used slightly different prompts for mathematical reasoning benchmarks compared to commonsense reasoning, emphasizing numbers to fully preserve the mathematical information in the solutions.

A.2 Question Generating Models' Generated Question

To evaluate the performance of the question generation model, we considered a total of three models: Llama3.2-1B, Qwen2.5-0.5B, and Flan-T5-base. As shown in Table 4, the Llama3.2-1B and Qwen2.5-0.5B models essentially failed to generate proper questions. In contrast, we observed that T5 was able to perfectly reconstruct questions corresponding to the given answers. Consequently, along with semantic similarity criteria and after observing these cases, we selected the T5 model.

A.3 Fail Case of SLMs in Conventional Method

In our experiments, as shown in Table 5, models under 1B parameters such as SmolLM2 and Llama experienced performance degradation when conventional feedback methods like USC and self-correction were applied. To help understand these situations, we have attached fail cases.

	Arithmetic reasoning	Commonsense reasoning
Solution	Marco's dad's strawberries weighed 11 pounds. Together they weighed 30 pounds. Since Marco's dad weighed 11 pounds, Marco must have weighed 30 - 11 = 19 pounds. The answer is 19.	Going to church is one way people can hear testimony of how spirituality changes lives. Therefore, the answer is (A).
Prompts	Do not change ANY numeric values in the answer. Every number must be preserved EXACTLY in your question. Generate a question that would have this as its answer:	From the commonsense reasoning answer provided below, recreate the original commonsense question. Generate a question that would have this as its answer:
Generated Questions	Marco and his dad went strawberry picking. Marco's dad's strawberries weighed 11 pounds. If together their strawberries weighed 30 pounds. How much did Marco's strawberries weigh?	You can hear testimony of how spirituality changes lives when you do what?

Table 3: Examples of question generating prompts. Input solutions and prompts for question generation in arithmetic reasoning and common sense reasoning.

Model	Generated Question
Original Answer	Janet's ducks lay 16 eggs per day. She eats 3 eggs for breakfast. That means she has 16 - 3 = 13 eggs left each day. She bakes muffins with 4 eggs and sells the rest for $2 each. That means she sells 13 - 4 = 9 eggs per day. She sells 9 eggs for $2 each, so she earns 9 * 2 = 18 dollars per day. The answer is 18.
LLaMA	Note: The problem is not a typical subtraction problem, but rather a problem that involves subtraction and then adding back the result. It is also not just a subtraction, it is a combination of subtraction (for the muffin sales) and adding the earnings from selling the eggs (from the remaining eggs).
Qwen	(Actually, Janet earns more than $10 because she can sell the remaining 5 eggs at $0.50 each.) Question: If Janet's ducks laid 640 eggs in a week, how much money would Janet make from selling the eggs if she eats them all up and then bakes for her family of x people? To solve this problem, let's break down the information given: 1. Janet has ducks that lay eggs. 2. Each duck lays...
T5	Janet's ducks lay 16 eggs per day. She eats 3 eggs for breakfast each day and bakes muffins with 4 eggs. How much money does she earn per day if she sells 9 eggs for $2 each?

Table 4: Comparison of Generated Questions by Different Models

Model	SmolLM2-360
Input	Prompt: Evaluate these responses. Select the most consistent response based on majority consensus. Start your answer with "The most consistent response is Response X" (without quotes) response1: Janet makes $2 per fresh duck egg. She eats three for breakfast every morning and bakes muffins for her friends every day with four. response2: Janet makes 16 / 3 = 6 eggs per day. 16 / 3 = 5 eggs per day. 3 / 5 = 0.6 eggs per day. 6 / 3 = 1.6 eggs per day. 1.6 / 5 = 0.34 eggs per day. ..... response5: She spends $2 / day * 16 eggs per day = $32. She makes $32 - $16 = $18. The answer is 18.
USC Output	"Response 1: The most consistent response is Response X.Response 2: The most consistent response is Response Y.Response 3: The most consistent response is Response Z.Response 4: The most consistent response is Response W.Response 5: The most consistent response is Response X.Response 6: The most consistent response is Response Y.Response 7: The most consistent response is Response Z.Response 8: The most consistent response is Response W.Response 9: The most consistent response is Response X.Response 10: The most consistent response is Response Y.Response 11: The most consistent response is Response Z.Response 12: The most consistent response is Response W.Response 13: The most consistent response is Response X.Response 14: The most consistent response is Response Y.Response 15: The most consistent response is Response Z.Response 16: The most consistent response is Response X.Response 17: The most consistent response is Response Y.Response 18: The most consistent","

Table 5: Model evaluation with USC method. It shows that SmolLM2 that cannot understand when long input context is provided