Better Zero-Shot Reasoning with Self-Adaptive Prompting

Xingchen Wan[*1,3], **Ruoxi Sun**[1], **Hanjun Dai**[2], **Sercan Ö. Arık**[1], **Tomas Pfister**[1]
[1]Google Cloud AI Research
[2]Google DeepMind
[3]Department of Engineering Science, University of Oxford
`{xingchenw,ruoxis,hadai,soarik,tpfister}@google.com`

- incontext learning (few shot prompting) poses a bigger challenge - LLMs are highly prone to even slight changes in the samples given as few shot prompts. this triggers the need for huge human effort in picking the correct examples for the prompts

- zero shot prompting does not guide the LLMs on tasks and hence task specific activities can be trickier

Abstract

Modern large language models (LLMs) have demonstrated impressive capabilities at sophisticated tasks, often through step-by-step reasoning similar to humans. This is made possible by their strong few and zero-shot abilities – they can effectively learn from a handful of handcrafted, completed responses (*"in-context examples"*), or are prompted to reason spontaneously through specially designed *triggers*. Nonetheless, some limitations have been observed. First, performance in the few-shot setting is sensitive to the choice of examples, whose design requires significant human effort. Moreover, given the diverse downstream tasks of LLMs, it may be difficult or laborious to handcraft per-task labels. Second, while the zero-shot setting does not require handcrafting, its performance is limited due to the lack of guidance to the LLMs. To address these limitations, we propose Consistency-based Self-adaptive Prompting (COSP), a novel prompt design method for LLMs. Requiring neither handcrafted responses nor ground-truth labels, COSP selects and builds the set of examples from the LLM *zero-shot outputs* via carefully designed criteria that combine consistency, diversity and repetition. In the zero-shot setting for three different LLMs, we show that using only LLM predictions, COSP improves performance up to 15% compared to zero-shot baselines and matches or exceeds few-shot baselines for a range of reasoning tasks.

1 Introduction

Recent advances in large language models (LLMs) have led to state-of-the-art performance in existing natural language processing (NLP) tasks (Wang et al., 2018, 2019; Thoppilan et al., 2022) and to exciting emergent abilities (Wei et al., 2022a). A prominent example of the latter is the strong performance on tasks that require analytical reasoning and/or methodical planning, which were previously thought to be difficult even for massive LLMs (Rae et al., 2021). This has been made possible by scaling the model size and training corpus, the strong few-shot and zero-shot abilities of modern LLMs, and new techniques such as the chain-of-thought (CoT) methods: *Few-shot CoT* (Wei et al., 2022b) prepends the test queries with solved input-output pairs as *in-context examples* to prompt the LLM to generate rationales. *Zero-shot CoT* (Kojima et al., 2022), on the other hand, appends trigger phrases after test queries to elicit reasoning. These methods constitute a competitive alternative to conventional fine-tuning which can be very costly for modern LLMs due to their sheer size, and is not possible when the LLMs are available as inference-only APIs – an increasingly popular paradigm for serving models like ChatGPT (Schulman et al., 2022).

Despite these promising advances, numerous open challenges remain. Although Zero-shot CoT is task-agnostic (and thus the most generally applicable in any scenario) and does not require human effort for labeling, it often underperforms its few-shot counterparts, with LLMs that are not shown with "template rationale" often producing spurious reasoning steps. On the other hand, for Few-shot CoT, the performance has been shown to be sensitive to the choice of the demonstrations (Wei et al., 2022b), and thus improving the performance might require significant trial-and-errors and/or specific relevant expertise. Given the fact that modern LLMs are often used for very diverse downstream tasks, selecting even a few useful per-task examples can quickly become more laborious as the number of tasks increase. This is particularly true for the CoT tasks where *both* rationales and answers need to provided in the demonstrations. In some other cases, it could be difficult or simply impossible even to handcraft a few examples beforehand: for example, there might be tasks involving tedious and complicated reasoning or planning and/or novel test-time tasks unseen previously.

[*]Work done during internship at Google.

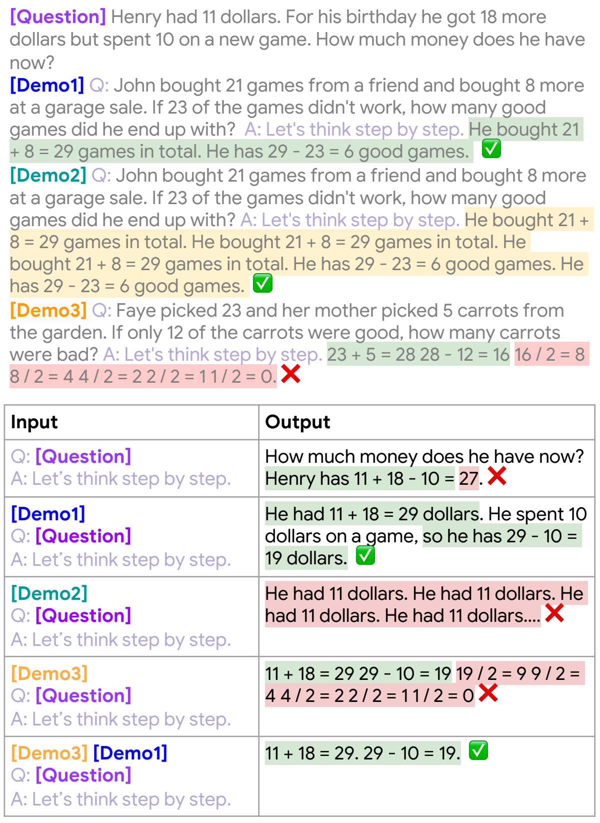
In this work, our focus is to improve LLM reasoning in the general zero-shot setup with access to input queries but not labels (i.e., *transductive* zero-shot (Xian et al., 2017)). We are particularly interested in this setup due to its aforementioned generality, but it is also a very challenging one given the lack of external guidance. To address this, instead of relying on simple triggers or handcrafted examples, we propose prompting the LLMs with the outputs *generated by their own*. To achieve this, we first collect a pool of rationales and answers to the test questions that are generated by the LLM using Zero-shot CoT, and then select the most suitable questions and answers for in-context learning, similar to Few-shot CoT. The choice of *which* self-generated rationales to include, however, is highly non-trivial, as illustrated in Fig. 1: not only does this problem inherit all of the challenges identified above for the few-shot scenario, it also carries the additional difficulty that the self-generated demonstrations themselves are imperfect. Retrieval methods assuming perfect labels (Rubin et al., 2022; Liu et al., 2022) are shown to perform worse than random in this setup (Zhang et al., 2022), and previous works bypass this issue by focusing only on surface text similarity and diversity of the question embeddings (Zhang et al., 2022), which as we show, in cases when most of the initial generated outputs are wrong, can be misleading.

- can be an interesing point as similar methods like active prompting also uses uncertainity (entropy based)for selection of prompts for few shot prompting which require labels

To address these challenges, we propose COSP (Consistency-based Self-adaptive Prompting), an algorithm requiring only unlabelled samples (which are typically cheap to obtain, for example, via continuous, on-the-fly collections of user queries) and no labelled examples for strong performance, while also providing the capability of further improving the performance when a few labelled examples become available. The algorithm consists of two stages. In the first stage, COSP collects the pool of LLM responses to test questions via Zero-shot CoT. We use these responses to compute the *outcome entropy* of each question, a metric inspired by *self-consistency* (Wang et al., 2022a). This metric is then used to identify suitable question-reasoning pairs in an unsupervised way.

In the second stage, we perform another pass over all test samples, but with the selected question-reasoning pairs as in-context demonstrations.

We experiment with COSP with three different LLMs on a range of tasks. We demonstrate a 10–15% improvement in average accuracy for 6 arithmetic and logical reasoning tasks over Zero-shot CoT with self-consistency baseline in PaLM-62B and GPT-3, and >3% improvement in PaLM-540B. All of these are achieved with a negligible additional computational cost. In many cases, COSP performs on par or better than the few-shot baselines with handcrafted in-context examples.



[Question] Henry had 11 dollars. For his birthday he got 18 more dollars but spent 10 on a new game. How much money does he have now?
[Demo1] Q: John bought 21 games from a friend and bought 8 more at a garage sale. If 23 of the games didn't work, how many good games did he end up with? A: Let's think step by step. He bought 21 + 8 = 29 games in total. He has 29 - 23 = 6 good games. ✅
[Demo2] Q: John bought 21 games from a friend and bought 8 more at a garage sale. If 23 of the games didn't work, how many good games did he end up with? A: Let's think step by step. He bought 21 + 8 = 29 games in total. He bought 21 + 8 = 29 games in total. He bought 21 + 8 = 29 games in total. He has 29 - 23 = 6 good games. ✅
[Demo3] Q: Faye picked 23 and her mother picked 5 carrots from the garden. If only 12 of the carrots were good, how many carrots were bad? A: Let's think step by step. 23 + 5 = 28 28 - 12 = 16 16 / 2 = 8 8 / 2 = 4 4 / 2 = 2 2 / 2 = 1 1 / 2 = 0. ❌

Input	Output
Q: **[Question]** A: Let's think step by step.	How much money does he have now? Henry has 11 + 18 - 10 = 27. ❌
[Demo1] Q: **[Question]** A: Let's think step by step.	He had 11 + 18 = 29 dollars. He spent 10 dollars on a game, so he has 29 - 10 = 19 dollars. ✅
[Demo2] Q: **[Question]** A: Let's think step by step.	He had 11 dollars. He had 11 dollars. He had 11 dollars. He had 11 dollars.... ❌
[Demo3] Q: **[Question]** A: Let's think step by step.	11 + 18 = 29 29 - 10 = 19 19 / 2 = 9 9 / 2 = 4 4 / 2 = 2 2 / 2 = 1 1 / 2 = 0 ❌
[Demo3] [Demo1] Q: **[Question]** A: Let's think step by step.	11 + 18 = 29. 29 - 10 = 19. ✅

Figure 1: *Selecting in-context demonstrations for reasoning tasks can be a delicate art.* LLM output is sensitive to in-context demos and their reasoning, especially when they are generated and imperfect. Example inputs & outputs shown from top to bottom (MultiArith dataset & PaLM-62B model): (1) zero-shot CoT with no demo: correct logic but wrong answer; (2) correct demo (Demo1) and correct answer; (3) correct but repetitive demo (Demo2) leads to repetitive outputs; (4) erroneous demo (Demo3) leads to a wrong answer, but (5) combining Demo3 and Demo1 again leads to a correct answer. This motivates the need for a carefully-designed selection procedure for in-context demos, which is the key objective of this paper.

2 Preliminaries

Chain-of-thought (CoT) Prompting. Few-shot CoT prompting (Wei et al., 2022b) prepends test questions with a series of in-context demonstrations of related solved questions as prompts. Generalizing beyond the few-shot setup, Kojima et al.

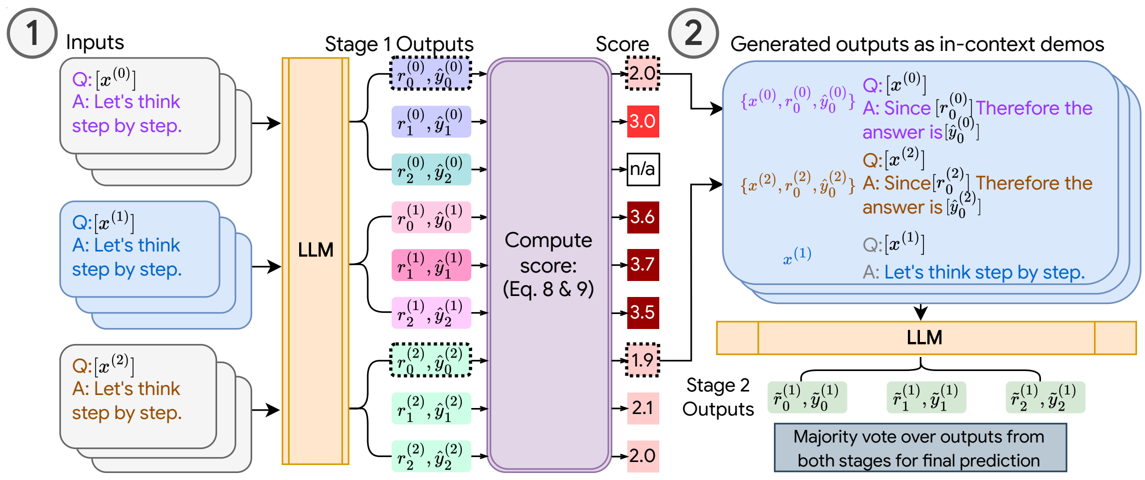

1
Inputs
Q: $[x^{(0)}]$
A: Let's think step by step.
Q: $[x^{(1)}]$
A: Let's think step by step.
Q: $[x^{(2)}]$
A: Let's think step by step.
LLM
Stage 1 Outputs
$r_0^{(0)}, \hat{y}_0^{(0)}$
$r_1^{(0)}, \hat{y}_1^{(0)}$
$r_2^{(0)}, \hat{y}_2^{(0)}$
$r_0^{(1)}, \hat{y}_0^{(1)}$
$r_1^{(1)}, \hat{y}_1^{(1)}$
$r_2^{(1)}, \hat{y}_2^{(1)}$
$r_0^{(2)}, \hat{y}_0^{(2)}$
$r_1^{(2)}, \hat{y}_1^{(2)}$
$r_2^{(2)}, \hat{y}_2^{(2)}$
Compute score: (Eq. 8 & 9)
Score
2.0
3.0
n/a
3.6
3.7
3.5
1.9
2.1
2.0
2
Generated outputs as in-context demos
$\{x^{(0)}, r_0^{(0)}, \hat{y}_0^{(0)}\}$
Q: $[x^{(0)}]$
A: Since $[r_0^{(0)}]$ Therefore the answer is $[\hat{y}_0^{(0)}]$
$\{x^{(2)}, r_0^{(2)}, \hat{y}_0^{(2)}\}$
Q: $[x^{(2)}]$
A: Since $[r_0^{(2)}]$ Therefore the answer is $[\hat{y}_0^{(2)}]$
$x^{(1)}$
Q: $[x^{(1)}]$
A: Let's think step by step.
LLM
Stage 2 Outputs
$\tilde{r}_0^{(1)}, \tilde{y}_0^{(1)}$
$\tilde{r}_1^{(1)}, \tilde{y}_1^{(1)}$
$\tilde{r}_2^{(1)}, \tilde{y}_2^{(1)}$
Majority vote over outputs from both stages for final prediction

Figure 2: The overall procedure of COSP. In *Stage 1* (**Left**, §3.2), we run Zero-shot CoT multiple times on each question to generate a pool of demonstrations (each consisting of the question $x^{(i)}$, generated rationale $r_j^{(i)}$ and predicted answer $\hat{y}_j^{(i)}$) and assign a score to each. Note that different rationales leading to the same final answer are marked the same color in "Outputs" blocks (described in §3.2). In *Stage 2* (**Right**, §3.3), we augment the current test question for which we are interested in finding in-context examples (boxes shaded in blue) with a number of in-context demonstrations. These consist of test questions and selected Stage 1 outputs *whose predicted answers* $\hat{y}_j^{(i)}$ *are the majority predictions for that question* that minimize the score defined in Eq. (8) with the procedure described in §3.2. The augmented test question is used to query the LLM a second time. Finally, a majority vote over outputs from both stages forms the final prediction.

(2022) propose *Zero-shot CoT* and shows that even by simply concatenating a trigger phrase (e.g., "Let's think step by step") to the test question, LLMs can be prompted to output intermediate steps in absence of handcrafted examples (we show an illustration of Few-shot and Zero-shot CoT in Fig. 5 in App. A). Our work builds upon on these works and improves upon them: we use the same zero-shot setup as Kojima et al. (2022), while enabling in-context learning in the format of Wei et al. (2022b) through an innovative use of the LLM's own outputs.

Self-consistency. Self-consistency (Wang et al., 2022a; Li et al., 2022b), which approximates the marginal distributions of the LLM via majority voting of multiple decoded reasoning paths, has been shown to lead to significant performance boosts across various tasks. It is compatible with either Zero- or Few-shot CoT by first introducing probabilistic decoding in the LLM decoding using a non-zero temperature, and then sampling m reasoning paths, each consisting a rationale-answer pair $(r^{(i)}, \hat{y}^{(i)})$ from the LLM posterior:

- self consistency technique is a voting technique.
- here in COSP, the same question is zero shot prompted to LLM multiple times.
- the rationales and labels are generated multiple times
- the label that is predicted most times is chosen as the final output

$$\{(r_j^{(i)}, \hat{y}_j^{(i)})\}_{j=1}^{m} \sim \mathbb{P}(r^{(i)}, \hat{y}^{(i)} | x^{(i)}, c, \theta), \quad (1)$$

where $x^{(i)}$ are the test questions, c denote the general format of the prompt template (e.g. the choice of in-context demonstrations in Few-shot CoT or the trigger phrase in Zero-shot CoT) and θ denote the weights that parameterize the LLM. The prediction with the *majority* (or more precisely, the *plurality*) vote is then chosen as the final prediction:

$$\hat{y}^{(i)} = \arg\max_{\hat{y}_j^{(i)}} \sum_{k=1}^{m} \mathbb{I}(\hat{y}_j^{(i)} = \hat{y}_k^{(i)}), \quad (2)$$

where $\mathbb{I}(\cdot, \cdot)$ is the indicator function. We use and extend self-consistency as an integral part of our algorithm. Instead of simply taking the majority vote, we also use it as a key criterion for selecting the in-context demonstrations, which we discuss in detail in §3.

3 Consistency-based Self-adaptive Prompting (COSP)

As discussed in the Introduction, our goal is to develop a method that improves zero-shot reasoning abilities of LLMs and reduces the need for human effort by automatically selecting in-context demonstrations from the LLM's own generated outputs. To effectively achieve this, we propose COSP (overall procedure shown in Fig. 2 and Algorithm 1).

In this section, we present the problem setup, followed by detailed descriptions of the two stages and the key components of the algorithm.

3.1 Notations & Problem Setup

What is a demonstration in COSP ?

- when a LLM is prompted with zero-shot settings and non-zero temperature for 'n' questions and 'm' times for each of 'n' question then we will get nm combinations of [q,r (rationale),y (predictions)]. this is called the candidate pool (p)

- next self consistency is used to select 'd' candidates from this pool (p) . these 'd' candidates are called demonstrations.

We assume a test set $\mathcal{D}$ with $|\mathcal{D}| = N$ test examples. Given an LLM and a test query $x^{(i)} \in \mathcal{D}$ (the test label, $y^{(i)}$ is not available to the LLM), we seek to generate a set of K demonstrations $\mathcal{S}$, where each demonstration s_k is a concatenation of another test question $x^{(i_k)} \in \mathcal{D}_{\setminus i}$, the generated rationale ($r^{(i_k)}$) and final prediction ($\hat{y}^{(i_k)}$) from the LLM (typically via Zero-shot CoT. The details are provided in §3.2):

$$s_k = \texttt{Concatenate}(x^{(i_k)}, r^{(i_k)}, \hat{y}^{(i_k)}). \quad (3)$$

The set $\mathcal{S}$ is then prepended as the "context" to the test question x_i, serving as a guide to the LLM:

$$\tilde{x}^{(i)} = \texttt{Concatenate}(s_1, ..., s_K, x^{(i)}), \quad (4)$$

and $\tilde{x}^{(i)}$ is queried again (i.e., *in-context learning*). The objective is to build the set of in-context demonstrations $\mathcal{S}$ that maximize the LLM performance without accessing the ground-truth labels.

3.2 Stage 1: Building the Pool of Generated Responses & the Demonstration Selector

Building the Candidate Pool. We first build a pool of responses $\mathcal{P}$ to the test questions $\mathcal{D}$ from which demonstrations $\mathcal{S}$ are selected. Each element of $\mathcal{P}$ is a candidate demonstration in the format defined in Eq. (3). To achieve this, we leverage the zero-shot capabilities of LLMs by running Zero-shot CoT (Kojima et al., 2022) over all test questions For each test question $x^{(i)}$, we query the LLM m times with non-zero temperature to extract multiple reasoning paths $\{r_j^{(i)}\}_{j=1}^m$ and potentially different answers $\{\hat{y}_j^{(i)}\}_{j=1}^m$ according to Eq. (1).

Sampling m paths per question yields mN candidates at this stage.

Difficulty in COSP :

- identifying the correct exemplars from the list of candidate pool (p)

- candidate pool rationales and responses are completely unsupervised.

Consistency for Demonstration Selection. With the candidate pool built, we now select $\mathcal{S}$ from it. This selection process is highly non-trivial – not only is in-context learning known to be sensitive to the choices of the demonstrations (Wei et al., 2022b; Liu et al., 2022), we also have the additional challenges that (1) we have to select K (typically small and ≤ 10) from a large set of candidates $\mathcal{P}$, and that (2) the candidate pool is imperfect: in the absence of access to ground-truth labels, the responses in $\mathcal{P}$ are potentially erroneous, and could be misleading to the LLM.

To address these problems, we propose to use *self-consistency* both (1) to prune the candidate pool $\mathcal{P}$, and (2) to select the demonstrations in absence of ground-truth labels. Drawing upon the insight from Wang et al. (2022a) that "*majority* predictions are more likely to be correct", for each test question $x^{(i)}$ we first compute their majority vote prediction(s) $\hat{y}^{(i)}$ from all predictions $\{\hat{y}_j^{(i)}\}_{j=1}^m$ with Eq. (2) and retain only the rationales that lead to the majority vote prediction (which are more likely to be correct) and prune the rest of the rationale-prediction pairs (which are more likely to be wrong) from $\mathcal{P}$. Following previous work (Wei et al., 2022b), we also use further heuristics, which we detail in App. B, to remove obviously bad candidates from $\mathcal{P}$ (e.g. responses containing no numbers for arithmetic tasks, or overly short and/or fragmented responses). Formally, after pruning, for test question $x^{(i)}$, the set $\mathcal{P}$ is given by:

- selection of q,r,y pair for each question q using majority predictions voting and heuristics

$$\mathcal{P} = \bigcup_{i=1}^{N} \bigcup_{j=1}^{\bar{m}^{(i)}} \left(x^{(i)}, r_j^{(i)}, \hat{y}^{(i)} \right), \quad (5)$$

where j indexes over all $\bar{m}^{(i)}$ reasoning paths in the i-th question that led to the majority vote prediction $\hat{y}^{(i)}$ and are not otherwise excluded.

The second usage of self-consistency draws upon the insight that it approximates the amount of uncertainty of the model for its prediction. It is conceptually related to *pseudo-labelling* (Lee et al., 2013; Shi et al., 2018; Rizve et al., 2021) and *entropy minimization* (Grandvalet and Bengio, 2004). Intuitively, if the LLM outputs the same prediction repeatedly even under different reasoning paths, it is expected to be more confident in its prediction, and we should assign a stronger belief that the prediction is correct or at least plausible ("*consistent* predictions are more likely to be correct"). To capture the model uncertainty, for a question $x^{(i)}$ with m final answers from which we assume that there are $n \leq m$ unique answers $\hat{y}_1^{(i)}, ..., \hat{y}_n^{(i)}$, we compute the *normalized entropy* as:

$$\mathbb{H}(x^{(i)} | \{\hat{y}_j^{(i)}\}_{j=1}^m) = \frac{\sum_{\alpha=1}^{n} \tilde{\mathrm{p}}(\hat{y}_\alpha^{(i)}) \log \tilde{\mathrm{p}}(\hat{y}_\alpha^{(i)})}{\log m}, \quad (6)$$

where α is used to index over the unique answers, $\tilde{\mathrm{p}}(\hat{y}_\alpha^{(i)})$ is the empirical frequency of unique answer

Algorithm 1 COSP. Stage 1 (§3.2) and Stage 2 (§3.3) steps marked in orchid and peach respectively. COSP⁺-specific steps (refer to §3.3) are marked in dark orange.

```
1: Input: Test questions $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$, LLM, # paths per
   sample $m$, Pool of generated responses $\mathcal{P} \leftarrow \emptyset$.
2: Output: Predictions $\{\hat{y}^{(i)}\}_{i=1}^{N}$.
3: for $i \in [1, N]$ do
4:     Query the LLM with Zero-shot CoT to obtain $m$ ratio-
       nales and predictions $\{r_j^{(i)}, \hat{y}_j^{(i)}\}_{i=1}^{m}$ to $x^{(i)}$.
5:     Add candidate demos (Eq. (3)) that led to the majority
       vote prediction to $\mathcal{P}$.
6: end for
7: Build the set of demonstrations $\mathcal{S}$ using the procedure
   described in §3.2 and Eq. (8) & (9).
8: for $i \in [1, N]$ do
9:     Concatenate the $\mathcal{S}$ to $x^{(i)}$ (Eq. (4)) and query the LLM
       again to obtain $m$ new rationales and answers (COSP)
       / an adaptive number of new rationales and answers
       proportional to the entropy of $x^{(i)}$ (COSP⁺).
10:    Compute the majority vote over $2m$ answers from both
       stages as the final answer $\hat{y}^{(i)}$.
11: end for
```

$\hat{y}_\alpha^{(i)}$ in all m answers, and the entropy is normalized by $\log m$, the (negative) maximum entropy if all predictions are different from each other, to $[0, 1]$.

As shown in Fig. 3, we indeed find that the normalized entropy is a good proxy over a number of different tasks where low entropy is positively correlated with correctness.

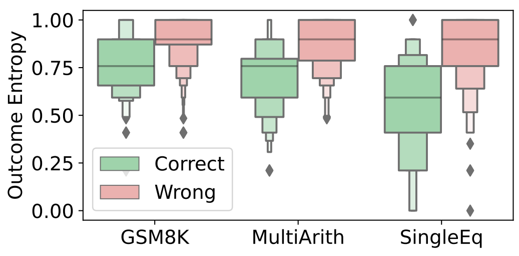


Figure 3: *Entropy over outcomes approximates correctness.* Distributions of entropy (Eq. (6)) vs. the ground-truth accuracy using Zero-shot CoT of the majority vote predictions out of 7 sampled paths on GSM8K, MultiArith and SingleEq using PaLM-62B (Chowdhery et al., 2022). Questions answered correctly have significantly lower entropy. Further analyses are provided in App. C.1.

Penalizing Repetitions. Eq. (6) scores the *test questions* $x^{(i)}$, but for each $x^{(i)}$, especially for those with a low outcome entropy, there are multiple *reasoning paths* $r_j^{(i)}$ for the majority prediction. While they lead to the same final answer, we find that their suitability as in-context demonstrations nonetheless differs. A key observation is that *repetitive demonstrations often lead to worse performance*, as exemplified in Fig. 1 (see `Demo2`). We hypothesize that repetitive responses acting as demonstrations create a strong but spurious pattern on which LLMs are prone to overfit, often leading to catastrophic deterioration of performance.

- it was observed that using repetitive demonstrations lead to poor outputs.refer fig1 - demo 2

- here repetitive means a candidate demonstration having repetitive phrases.

- phrases are split using regex heuristics and a similarity matrix is constructed using cosine similarity

To address this, we first filter out the responses that involve self-questioning which we observe often leads to repetition or prompts the LLM to answer the generated questions instead of the original questions. We also introduce a quantitative measure by splitting demonstrations into phrases delimited by punctuation marks (“`[.,?!]`”). Assuming that we have Q phrases[*], we compute repetitiveness as:

$$R_r(r_j^{(i)}) = \frac{2}{Q(Q-1)}\Big(\sum_{a=1}^{Q}\sum_{b=a+1}^{Q} W_{ab}\Big), \quad (7)$$

where $\mathbf{W} \in \mathbb{R}_{+}^{Q\times Q}$ is the similarity matrix over all pairs of phrases with $W_{ab} = S_c(\phi(q_a), \phi(q_b))$, where $S_c(\cdot,\cdot)$ computes the cosine similarity and $\phi(q_a)$ and $\phi(q_b)$ denote the vector embedding of a-th and b-th phrases, typically obtained through an auxiliary, small language model. From Eq. (7), the matrix is summarized by averaging across off-diagonal elements, which gives an average similarity between two arbitrary phrases in the response. We note that it is possible for other techniques that penalize repetition at decoding-time to be applied in lieu (like hard or soft n-gram blocking (Klein et al., 2017; Paulus et al., 2017)). We use the proposed method as it also captures semantic-level repetitions, which we find to be more common than verbatim repetitions, especially when a non-zero temperature is used.

Overall Design & Building $\mathcal{S}$. With measures of consistency (Eq. (6)) and repetitiveness (Eq. (7)) defined, for a candidate demonstration $p \in \mathcal{P}$ given by the concatenation of question $x^{(i)}$, rationale $r_j^{(i)}$ and the (majority) prediction $\hat{y}^{(i)}$, the score is proposed as:

$$\mathcal{F}(p|x^{(i)}, r^{(i)}, \{\hat{y}_j^{(i)}\}_{j=1}^{m}) = \mathbb{H}(x^{(i)}) + \lambda R_r(r_j^{(i)}), \quad (8)$$

where λ is a trade-off hyperparameter that is set to 0.2 in all experiments. To ensure that the different terms of Eq. (8) are of a comparable scale, we

[*]Repetitiveness is undefined when $Q = 1$. In this case, we assign the minimum self-repetition score over all demonstrations where it is defined.

- scoring used to identify the final list of 'K' number of final demonstrations (S).

- for scoring, pruning (based on voting), heuristics and penalization of repetitions logics are used

Algorithm 2 Building $\mathcal{S}$ for $K \geq 2$.

```
1: Initialize $\mathcal{S}$ with the minimizer of Eq. (8): $\mathcal{S} \leftarrow \{p_0^* = \arg\min_{p \in \mathcal{P}} \mathcal{F}(p)\}$
2: for $k \in [2, K]$ do
3:     Find the minimizer of the modified objective (Eq. (9)): $p_k^* = \arg\min_{p \in \mathcal{P}} \mathcal{G}_k(p)$.
4:     Add $p_k^*$ to $\mathcal{S}$ and remove $p_k^*$ from candidate pool $\mathcal{P}$.
5: end for
```

employ z-score normalization commonly used in reinforcement learning (van Hasselt et al., 2016) and NLP (Deng et al., 2022) by replacing each term with their respective z-score, with the mean and standard deviation computed over $\mathcal{P}$. To select a single in-context demonstration ($K = 1$), we utilize the minimizer of the scoring function $p^* = \arg\min_{p \in \mathcal{P}} \mathcal{F}(p)$. More commonly, though, we would like to select multiple demonstrations, leading to a combinatorial selection problem. For computational feasibility, we use a greedy forward selection procedure (Caruana et al., 2004) as described in Algorithm 2. We initialize $\mathcal{S}$ with p^* and in a greedy way and select the minimizer of the following modified objective function:

$$\mathcal{G}_k(p) = \mathcal{F}(p) + \lambda R_q(p, \mathcal{S}_{k-1}) \; \forall k \in [2, K], \quad (9)$$

where $\mathcal{S}_{k-1}$ is the partially built demonstration set $\mathcal{S}$ with $k-1$ elements already selected, and

$$R_q(p, \mathcal{S}_{k-1}) = \max\Big(\{S_c(\phi(p), \phi(s_{k'}))\}_{k'=1}^{k-1}\Big) \quad (10)$$

is a term to encourage diversity in question types and reasoning patterns in $\mathcal{S}$ by penalizing demonstrations that are similar to previous ones. We additionally constrain that $\mathcal{S}$ should not contain multiple reasoning paths to the same question. The algorithm is repeated until all K demonstrations of $\mathcal{S}$ are selected.

3.3 Stage 2: Query with Generated In-context Demonstrations

With $\mathcal{S}$ selected, in Stage 2, we concatenate and prepend the demonstrations to the test question (Eq. (3) & (4)) and query the LLM again with m more repetitions. The final prediction for each question is then output as the majority vote across the predictions from both stages.

Outcome Entropy as a Proxy to Difficulty. We use the entropy (Eq. (6)) to select demonstrations in §3.2. However, it is also a natural gauge of *difficulty* of a test question to the LLM, as a higher entropy (thus a higher uncertainty) implies that the LLM may require additional demonstrations for this question. To this end, we also experiment with a variant of COSP termed COSP⁺ (Algorithm 1) in §5 which features an adaptively allocated number of in-context demonstrations per question that is proportional to its zero-shot entropy in Stage 1, with higher-entropy questions given more demonstrations. It is worth noting that while it is easy to select more demonstrations in COSP as we only have to greedily optimize (8) until a higher K is reached, for methods like Few-shot CoT, where the number of demonstrations is also the number of "shots", increasing it requires handcrafting more completed examples.

Extension to Few-shot. While we have built COSP in the zero-shot setup, it can also be adapted to the few-shot setup (which we term COSP-FS) where, for example, a small number of labelled demonstrations $\mathcal{Q}$ are available, and we seek to augment them with more demonstrations without further manual labelling. The key modifications are: (1) instead of querying LLMs with *zero-shot* CoT in Stage 1 (Line 4 in Algorithm 1), we use *Few-shot CoT* with $\mathcal{Q}$; and (2) during construction of $\mathcal{S}$ (§3.2) we initialize with $\mathcal{Q}$ instead of an empty set. We experiment with COSP-FS in §5.

- COSP-FS uses few shot examples rather than using zero shot setting (like COSP) to prompt the LLMs to come up with the demonstration

- also when finalising the demonstrations, we start with empty list in case of COSP, whereas we start with the initial set of fewshot prompts in case of COSP-FS

Remarks on the Test Set Size. It is worth noting that while we have assumed the entire test set is available to COSP and perform experiments under that assumption (as we will show in §5) for straightforward comparison with the baselines, it is possible for COSP to work with only a fraction of test data or in an online mode where the unlabeled data are only collected continuously. One may easily achieve so by using the available data to perform Stage 1 of the COSP procedure. In Stage 2, we may perform majority voting over both stages only if the test query was available at the start of Stage 1, and majority voting *over Stage 2 only* is instead performed otherwise.

4 Related Works

There have been several recent works that also aim to improve LLM reasoning using the models' own outputs. STaR (Zelikman et al., 2022) is one of the first works to bootstrap reasoning from LLMs: starting from a small number of labelled data, STaR prompts the LLMs to generate rationales over the large corpus of untrained data. However, STaR requires human verification on the generated ratio-

Model	**PaLM-62B**					**PaLM-540B**				
Setting	**0-shot**			**5-shot**	***Prev***	**0-shot**			**5-shot**	***Prev***
Method	0-shot CoT	Auto-CoT	COSP *(Ours)*	5-shot CoT	*8-shot CoT*	0-shot CoT	Auto-CoT	COSP *(Ours)*	5-shot CoT	*8-shot CoT*
# Paths	14	14	7+7	14		14	14	7+7	14	
MultiArith	67.2	9.4	**85.0**	**81.0**	-	95.2	**99.0**	**98.8**	96.0	*99.3*[b]
AddSub	69.1	**73.2**	**78.9**	72.4	-	88.9	**89.1**	**89.9**	86.6	*93.7*[b]
SingleEq	74.4	77.8	**78.7**	**79.8**	-	88.6	85.6	**90.4**	**89.2**	-
GSM-8K	20.9	9.2	**30.2**	**30.3**	*27.4*[a]	68.5	71.4	**71.9**	64.3	*74.4*[b]
CSQA	46.5	**68.2**	60.2	**66.8**	-	74.2	**79.4**	76.4	**80.7**	*80.7*[b]
StrategyQA	57.2	59.4	**64.7**	**67.9**	-	66.0	**75.7**	75.2	**81.4**	*81.6*[b]
(Average)	55.88	49.55	**66.28**	**66.37**	-	80.25	**83.37**	**83.77**	83.03	-

[a]Madaan and Yazdanbakhsh (2022). [b]Wang et al. (2022a): Significantly more (**40**) paths sampled.

Table 1: Accuracy on PaLM-62B (**Left**) and PaLM-540B (**Right**) (Chowdhery et al., 2022). *Methods in the **0-shot** columns use no ground-truth label guidance*, whereas the **5-shot** results use 5 manually labelled in-context demonstrations, randomly selected from the provided examples in Wei et al. (2022b). ***Prev*** columns show results previously published under similar setups but often with significantly more sampling paths and/or labelled examples. "# Paths" denotes the number of reasoning paths sampled per test question. COSP uses $m = 7$ paths per stage and $2m = 14$ paths in total. We also report baseline results *without self-consistency* (which are worse) in App. C.2. COSP and Auto-CoT use 5 generated in-context examples per test question. The top two results for each model are bolded and ranked by color: **best** and **second-best**.

nale. More recently, Huang et al. (2022) relies on self-consistency to improve LLMs by fine-tuning on a large number of generated questions and rationales. However, both works rely on fine-tuning of massive models, which requires access to model weights and is much more expensive. In contrast, our method can treat the model as a black box and only requires forward passes. Other concurrent related works include Honovich et al. (2022) and Wang et al. (2022b). However, while related methodologically to our method, these papers focus on fundamentally different problems of data generation and instruction following.

For the in-context learning setup (Brown et al., 2020; Wei et al., 2022a; Radford et al.), various methods have been proposed, typically to retrieve relevant demonstrations from a relatively large number of labelled examples (Rubin et al., 2022; Su et al., 2022). However, Zhang et al. (2022) shows that with imperfect labels, nearest neighbor-style retrieval underperforms random selection and propose Auto-CoT, one of the few existing works that are fully comparable to COSP. Auto-CoT also uses models' zero-shot outputs as in-context demonstration by selecting the responses to the questions that are the k-means centroids in the embedding space from an auxiliary language model (a concurrent work, Li et al. (2022a) uses a similar clustering approach). However, by selecting based on *question* embedding only, Auto-CoT has no control over the quality of the generated *rationales*, and is thus prone to generating erroneous and misleading demonstrations; we empirically compare to it in §5. Another concurrent work (Weng et al., 2022) proposes *self-verification*, which is a zero-shot reasoning verifier to recover conditions from the predictions. However, it only works when there is a unique correct condition that results in the correct answer (such as arithmetic questions), but is not applicable when there are multiple plausible conditions.

5 Experiments

5.1 Experimental Setup

Datasets & Tasks. We consider the following arithmetic and logical reasoning tasks: AddSub (Hosseini et al., 2014), MultiArith (Roy and Roth, 2016), GSM-8K (Cobbe et al., 2021), SingleEq (Koncel-Kedziorski et al., 2015), CSQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021). The licenses of these datasets are listed in App. B.

LLMs. We run experiments with 3 LLMs: PaLM with 62 billion parameters (PaLM-62B) and 540 billion parameters (PaLM-540B) (Chowdhery et al., 2022) (both quantized to `int8` precision) and GPT-3 with 175 billion parameters (`code-davinci-001`) (Brown et al., 2020; Chen et al., 2021). The readers are referred to App. B for further details.

Model	GPT-3 (`code-davinci-001`)				
Setting	**0-shot**			**5-shot**	***Prev***
Method	0-shot CoT	Auto-CoT	COSP (*Ours*)	5-shot CoT	*8-shot CoT*
# Paths	14	14	7+7	14	
MultiArith	50.3	**78.5**	**80.7**	60.7	*82.7*[b]
AddSub	43.5	31.9	**61.5**	**63.3**	*67.8*[b]
SingleEq	48.8	58.1	**64.8**	**65.9**	-
GSM-8K	**10.2**	6.6	8.7	**16.7**	*23.4*[b]
CSQA	29.2	50.9	**55.4**	**53.0**	*54.9*[b]
StrategyQA	47.8	**55.8**	52.8	**55.4**	*61.7*[b]
(Average)	38.32	46.96	**53.98**	**52.60**	-

[b]Wang et al. (2022a): Significantly more (**40**) paths sampled.

Table 2: Accuracy on GPT-3 (Brown et al., 2020; Chen et al., 2021). Refer to Table 1 for further explanations.

Baselines. We compare against Zero-shot CoT (Kojima et al., 2022) with and without self-consistency (Wang et al., 2022a), and Auto-CoT (Zhang et al., 2022). Auto-CoT does not use self-consistency originally but for a fair comparison and given that self-consistency significantly improves LLM performance, we also include an Auto-CoT baseline augmented with self-consistency (the results of original Auto-CoT without self-consistency, which have worse performance, are reported in App. C.2). Besides the zero-shot baselines, we also include Few-shot CoT (Wei et al., 2022b) with self-consistency as a comparison, where we use 5 handcrafted demonstrations randomly chosen from the provided lists of examples in Wei et al. (2022b) for all datasets to match the 5 generated demonstrations in COSP and Auto-CoT. For COSP, we sample $m = 7$ reasoning paths in the Stage 1 and Stage 2, respectively, and for Zero-shot CoT, Few-shot CoT and Auto-CoT experiments with self-consistency, equivalently we use 14 reasoning paths for each question. In addition to the reproduced baselines, we also reference previously published baselines under ***Prev*** columns in Tables 1 & 2 that use more labelled examples and/or sampling paths, both of which lead to further performance gains but at an additional cost. We show in §5.2 that COSP may also take advantage of more sampling paths when resources are available. As mentioned in §3.2, we use $\lambda = 0.2$ (Eq. (8)) for all experiments *without further hyperparameter tuning* – this value is chosen to reflect our a-priori belief of the relative importance of the objective function terms. While it is possible to achieve even further performance improvements by tuning this term, it is not investigated in the present work for a fair comparison against the baselines.

5.2 Results

Discussions on Main Results. We show the results on PaLM-62B, PaLM-540B and GPT-3 in Tables 1, and 2, respectively, and give some examples of the found demonstrations in App. D. Since we find that self-consistency improves performance in all cases, we only report the results with self-consistency in these tables; the readers are referred to App. C.2 for results without self-consistency. In all cases, COSP delivers improvements over Zero-shot CoT baselines with particularly large gains seen on PaLM-62B and GPT-3, achieving 10% $\sim$ 15% average improvement over the Zero-shot CoT with self-consistency baseline. Remarkably, even though COSP operates under a much more challenging setup without ground-truth labels, COSP performs on par or better than Few-shot CoT in almost all tasks that use labels. We argue the large gains seen in the smaller models that significantly reduced their performance gap to the very large models (e.g. PaLM-540B), can be particularly practically impactful, given the former's strong advantages in computational costs and general accessibility. Furthermore, we find COSP to be robust by consistently improving over the zero-shot CoT baseline in all but one case, whereas Auto-CoT sometimes deteriorates the performance. We perform a failure analysis in App. C.5, and show that by using self-consistency as a proxy of correctness, in most cases COSP is capable of identifying outputs with sound reasoning as in-context examples even when the task is very challenging to the LLM under the zero-shot setup. In contrast, Auto-CoT is more prone to selecting erroneous examples; while Zhang et al. (2022) claims these errors minimally affect performance, we find their effect to be very model and task-specific, holding true for larger models (e.g. PaLM-540B) and easier tasks. However, on more difficult tasks or smaller models, we find erroneous demonstrations to often lead to performance deterioration.

- autoCoT at times selects wrong examples.

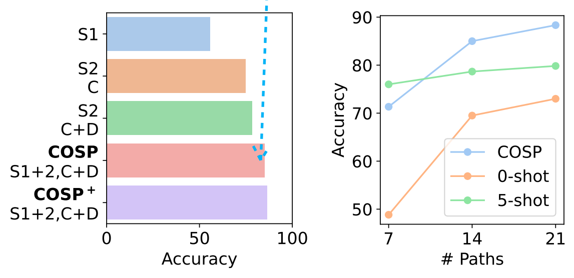
- though author of autoCoT claims this will impact final result but it depends on task and model size as easier tasks requiring less reasoning and bigger models means better reasoning capabilities

COSP⁺. We test COSP⁺ on the PaLM-62B model (see App. C.3). In 5 out of 6 cases, COSP⁺ leads to further performance gain. Across experiments, we use a fixed K which is shown to perform well overall, while COSP⁺ results show promise in *adaptively* setting K; we defer a thorough investigation to future work.

-but the performance gains were not much impactful

COSP-FS. We test COSP-FS in the few-shot setting as described in §3.3. We focus on GSM-

- this shows self-consistency with zero shot prompting is important to boost the performance gains



(a) Components of COSP. (b) Effect of # paths

Figure 4: Ablation results on MultiArith dataset using the PaLM-62B LLM.

8K, the most challenging task for PaLM-62B and GPT-3 where we hypothesize that labelled samples would help more. We start with 5 labelled examples provided in Wei et al. (2022b) in Stage 1. In Stage 2, we use COSP-FS to select 3 further examples. We show the results in Table 5 in App. C.4. It is worth noting that while in Table 2 COSP reduced the performance compared to Zero-shot CoT due to the difficulty of this task and selecting correct demonstrations from the zero-shot outputs (Table 7, App. D), the addition of labelled examples restored the effectiveness of COSP, primarily through the restoration of the predictive value of consistency (Eq 6) to correctness. We analyze this in App. C.5.

Ablation Studies. We perform ablation studies on COSP, using the PaLM-62B model on the MultiArith dataset. We first analyze the relative importance of various components of COSP and different terms in the scoring function Eq. (8) & 9. As shown in Fig. 4a, the results in the following settings demonstrate that all components of COSP positively contribute to the performance:

- zero shot with self consistency

1. `S1`: Majority vote on Stage 1 outputs only. This is essentially Zero-shot CoT over m paths.

- no consistency in stage1, consistency in stage2 and only entropy for scoring for selecting demonstrations

2. `S2,C`: Majority vote on Stage 2 outputs only (i.e. the m predictions from the LLM *after* in-context learning with $\mathcal{S}$). Only the entropy term (Eq. (6)) is used as the scoring function.

- no consistency in stage1, consistency in stage2, full scoring function

3. `S2,C+D`: Majority vote on Stage 2 outputs only with the full scoring function (Eq. (8) & 9).
4. **`COSP`**`(S1+2,C+D)`: Full COSP with majority votes over predictions from both stages.
5. **`COSP`**$^+$`(S1+2,C+D)`: COSP$^+$ from §3.3. Same as (iv) but with an adaptive number of demonstrations per sample.

Another possible baseline is to select demonstrations randomly from the Stage 1 outputs instead of using any demonstration selection heuristic described in §3.2; we do not include this as this is included in and outperformed by AutoCoT in Zhang et al. (2022) – we outperform even AutoCoT as shown in Tables 1 and 2. Lastly, we also analyze the effect of the number of sampling paths in Fig. 4b, where it is clear that the performance of all methods increases with number of paths. This demonstrate that there is potential for further performance gain by scaling COSP when additional computational resources are available.

6 Conclusion

We propose COSP, a prompting algorithm to improve zero-shot reasoning abilities of LLMs. It achieves this by selecting in-context demonstrations from its own outputs using a novel scoring function that incorporates consistency, diversity and repetitiveness. Across multiple LLMs and tasks requiring complex reassigning, we demonstrate large improvements over the state of the art. We believe there are multiple promising directions for future work: first, the present work focuses on improving zero-shot *reasoning*, but we argue that the key principle used in this paper may also be extended to more general NLP task types. Second, beyond solving reasoning problems *per se*, LLMs' reasoning capabilities are increasingly used in other scenarios for planning & interactions with external environments (Yao et al., 2023) – extending COSP to these setups could also be promising. We defer thorough investigations to a future work.

7 Limitations

One limitation is that the efficacy of our method is shown with massive LLMs in this paper. However, we note that our method is based on only model inference with them and is already considerably cheaper than other adaptation methods.

Furthermore, our method seeks to improve LLMs: while the technology itself is ethically neutral, we acknowledge that there are various social and ethical risks of potential misuse, especially given the powerful generative capabilities of the LLMs that have become increasingly accessible to a broader audience (Weidinger et al., 2021). We argue that both the prospective end-users and researchers should be aware of these concerns when using our method in order to mitigate these risks.

Methodologically, we note that an integral component of our algorithm is self-consistency. We rely on the expectation that it reliably predicts accuracy, which essentially places an expectation that the model uncertainty should be reasonably well-calibrated. While we have indeed found this to be the case for almost all considered tasks and models, additional investigations might be required to ascertain their general applicability. Given our reliance on self-consistency, for tasks where self-consistency does not lead to significant gains, the performance improvements with our method may be limited. An example of this could be tasks with very small label spaces (e.g. binary classification) where "consistency" in outcomes may be achieved much more easily even if the model simply outputs random predictions. A potential remedy, for which we defer a thorough investigation to future work, is to not only consider the consistency over outcomes but also over the intermediate rationales which are generated but not currently used for self-consistency computation.

Second, while COSP improves performance in an overwhelming majority of cases and is significantly less sensitive to the original zero-shot model performance compared to baseline methods like AutoCoT, there still exist exceptional cases where it fails to improve, especially when the tasks are too challenging in zero-shot setup – we argue that this is also due to the general, inherent limitations of the LLMs. However, both continual improvements on the foundational models and provision of some human guidance (e.g., using COSP-FS) should alleviate this issue.

References

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

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. 2004. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18.

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

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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*, 9:346–361.

Yves Grandvalet and Yoshua Bengio. 2004. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems (NeurIPS) 35*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896.

Junlong Li, Zhuosheng Zhang, and Hai Zhao. 2022a. Self-prompting large language models for open-domain qa. *arXiv preprint arXiv:2212.08635*.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2022b. On the advance of making language models better reasoners. *arXiv preprint arXiv:2206.02336*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Aman Madaan and Amir Yazdanbakhsh. 2022. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Mamshad Nayeem Rizve, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. 2021. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. *arXiv preprint arXiv:2101.06329*.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

J Schulman, B Zoph, C Kim, J Hilton, J Menick, J Weng, JFC Uribe, L Fedus, L Metz, M Pokorny, et al. 2022. Chatgpt: Optimizing language models for dialogue.

Weiwei Shi, Yihong Gong, Chris Ding, Zhiheng MaXiaoyu Tao, and Nanning Zheng. 2018. Transductive semi-supervised deep learning using min-max features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 299–315.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. 2022. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Hado P van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. 2016. Learning values across many orders of magnitude. *Advances in neural information processing systems*, 29.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the*

2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022a. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022b. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems (NeurIPS)*.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

Yixuan Weng, Minjun Zhu, Shizhu He, Kang Liu, and Jun Zhao. 2022. Large language models are reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. 2017. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4582–4591.

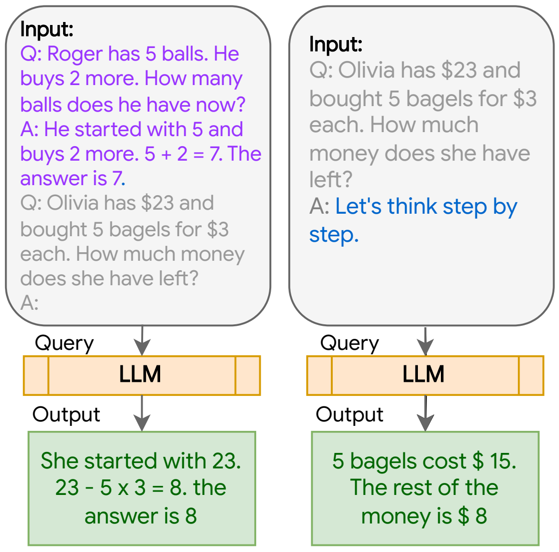
Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *International Conference on Learning Representations*.

Eric Zelikman, Jesse Mu, Noah D Goodman, and Yuhuai Tony Wu. 2022. Star: Self-taught reasoner bootstrapping reasoning with reasoning. *Advances in neural information processing systems*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

A Illustrations of Zero-shot and Few-shot CoT

We show additional illustrations of CoT prompting techniques described in §2 in Fig. 5.



Input:
Q: Roger has 5 balls. He buys 2 more. How many balls does he have now?
A: He started with 5 and buys 2 more. 5 + 2 = 7. The answer is 7.
Q: Olivia has $23 and bought 5 bagels for $3 each. How much money does she have left?
A:
Query
LLM
Output
She started with 23. 23 - 5 x 3 = 8. the answer is 8
Input:
Q: Olivia has $23 and bought 5 bagels for $3 each. How much money does she have left?
A: Let's think step by step.
Query
LLM
Output
5 bagels cost $ 15. The rest of the money is $ 8

Figure 5: Few-shot (**left**) and zero-shot (**right**) CoT. Both prompts the LLM to output intermediate steps: in few-shot (1-shot shown in the figure, with one provided completed question-rationale-answer set), handcrafted in-context examples are provided and prepended to the test questions. In the zero-shot setting, chain-of-thought output is prompted by *trigger phrases* ("Let's think step by step." in the figure).

B Additional Implementation Details

In this section, we describe the additional implementation details for the experiments: for the PaLM models, we follow Wang et al. (2022a) to use a temperature of 0.7 during decoding to extract different reasoning paths when self-consistency is used. On GPT-3 model, we use a temperature of 0.4 (for experiments without self-consistency, we use the default greedy decoding procedure with a temperature of 0). Both COSP and Auto-CoT involve the use of an auxiliary language model to extract sentence embeddings of questions and/or rationales. We use Sentence-T5-large (Ni et al., 2022) for COSP. It is worth noting that Auto-CoT (Zhang et al., 2022) originally uses SentenceBERT (Reimers and Gurevych, 2019); for consistency we use Sentence-T5 for Auto-CoT as well. Given that Ni et al. (2022) show that Sentence-T5 outperforms SentenceBERT for most sentence-level tasks, there should only be a positive impact on the baseline performances as a result of this change. For all tasks except for GSM-8K, we set a maximum decoding step of 128 tokens. On GSM-8K, we found that often long responses are generated and we increase the maximum decoding step to 256 tokens.

We also describe the additional heuristics used to prune the pool of candidates as described in §3.2. For demonstration selection, we reject the following questions or rationales which are observed to be inappropriate for our goals:

(i) Responses that feature no numbers (implemented via regular expression matching) for arithmetic tasks.

(ii) Responses that involve generated questions. While sometimes useful, we find that in general self-questioning is confusing, and often mislead the LLMs to answer the generated questions instead of the actual questions being asked.

(iii) Overly short rationales: responses containing fewer than 5 tokens.

(iv) Overly fragmented responses: after separating the rationales into phrases with punctuation marks as delimiters ("`[?!.,]`"), we filter out the responses that feature more than 10 fragments. This is similar to the heuristic used in Wei et al. (2022b) and Zhang et al. (2022) that remove overly long questions and responses.

We also conduct post-processing of the LLM outputs for all methods by, for example, cutting off outputs at stop tokens like "`\n\n\n`" and "`Q:`" (the latter is to prevent the LLM from generating further questions). For other pre- and post-processing steps and for answer extraction procedures, we follow Kojima et al. (2022) available at `https://github.com/kojima-takeshi188/zero_shot_cot`.

For the datasets, GSM-8K, CSQA, StrategyQA are licensed under the MIT license (GSM-8K: `https://github.com/openai/grade-school-math/blob/master/LICENSE`; CSQA: `https://github.com/jonathanherzig/commonsenseqa/issues/5`; StrategyQA: `https://github.com/eladsegal/strategyqa/blob/main/LICENSE`). MultiArith, SingleEq and AddSub datasets are licensed under the CC:BY 4.0 license (`https://www.cs.washington.edu/nlp/arithmetic`). We note that our use of these assets are consistent with their intended use (for research and developments), and the use of them for the purpose of evaluating LLMs is well-documented in the literature. For all datasets, we use the same data split as previous works like Kojima et al. (2022).

Since no training data is used, we only use the test split (when the test labels are publicly available) or the validation/dev split (otherwise).

C Additional Results & Analyses

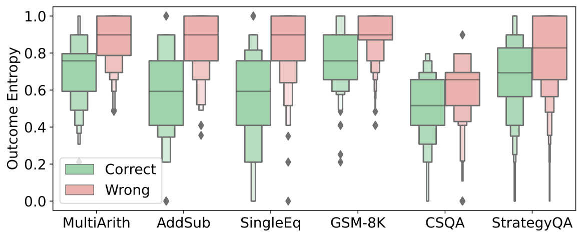
C.1 Outcome Entropy vs Correctness

Complementary to Fig. 3, in this section we provide additional experiments that demonstrate the link of the outcome entropy (Eq. (6)) and ground-truth correctness over more tasks and models. We show the full results in Fig. 6. We show that for a vast majority of cases, the outcome entropy is correlated with correctness in a statistically significant way (in all cases except for StrategyQA on GPT-3, the p-value testing the significance of the Point-biserial correlation (pbc) coefficient between entropy and accuracy is below 10^{-5}. The pbc coefficient on StrategyQA/GPT-3 is 0.04), with correctly answered questions featuring lower entropy. We find a stronger separation in entropy distribution in arithmetic tasks compared to the logic reasoning tasks (i.e., CSQA and StrategyQA). We argue that this is partly due to logic reasoning tasks considered in the paper being feature much smaller label spaces – while the answer to the arithmetic tasks can be any real number (essentially an infinitely-large label space), CSQA and StrategyQA feature multiple choice questions with 5 (choices A-E) and 2 (Yes/No) classes only, respectively. This is consistent with our discussions in §7 where we argue that the outcome entropy is more useful for large label space where it is very unlikely for random guessing to lead to correct answers, as opposed to classification tasks with few possible labels. As discussed in §7, a promising future direction is to address this issue and to improve the effectiveness of consistency on these types of tasks.

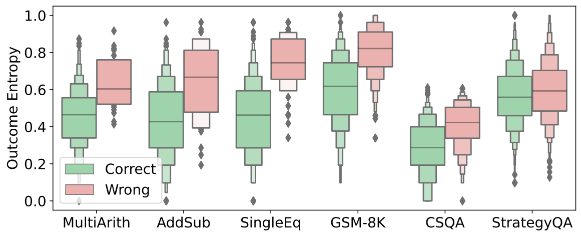
We further investigate the impact of the in-context demonstrations on the entropy by separating and comparing between Stage 1 and Stage 2 entropy and we show the comparison in Fig. 7, and we find that along with the improvement of test accuracy as reported §5, the addition of the demonstrations significantly also reduces the outcome entropy across all tasks and models.

C.2 Results without Self-consistency

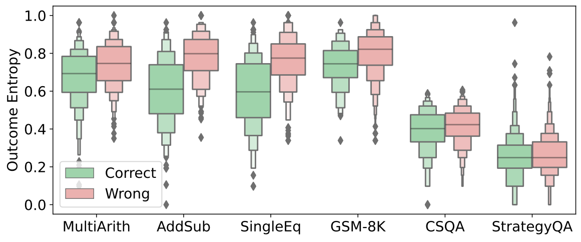
In this section, we report baseline results on Zero-shot CoT, Few-shot CoT and Auto-CoT without self-consistency on PaLM models, which are omitted in Table 1 in the main text. As shown in Table 3, we find self-consistency to increase performance across all models and tasks considered; due to this, we always use self-consistency in COSP, and only perform experiments with self-consistency in the GPT models.



(a) PaLM-62B



(b) PaLM-650B



(c) GPT-3

Figure 6: Distribution of entropy (Eq. (6)) vs ground-truth correctness using Zero-shot CoT for more tasks on PaLM and GPT-3.

C.3 COSP^{+} Results

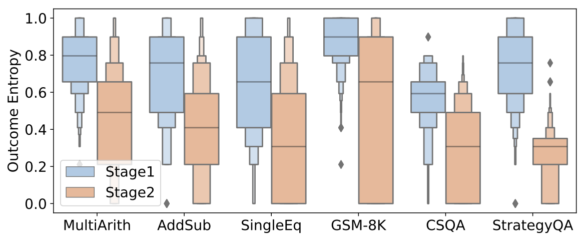
We perform preliminary experiments of COSP^{+} on PaLM-62B model, and we report the results in Table 4. In 5 out of 6 tasks, COSP^{+} leads to small improvements but it deteriorates the performance on GSM-8K as the additional demonstrations in this case contain more errors and became misleading to the LLM. We argue that the results show promise in more *adaptively* selecting demonstrations, and we defer a thorough exploration in a future work.

C.4 COSP-FS Results

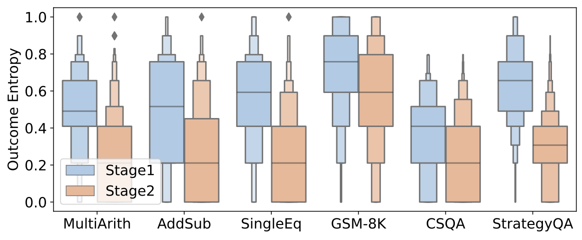
We compare COSP-FS against Few-shot CoT with self-consistency using PaLM-62B and GPT-3 on GSM-8K in Table 5.

Model	**PaLM-62B**			**PaLM-540B**		
Setting	**0-shot**		**5-shot**	**0-shot**		**5-shot**
Method	0-shot CoT	Auto-CoT	5-shot CoT	0-shot CoT	Auto-CoT	5-shot CoT
# Paths	1	1	1	1	1	1
MultiArith	24.5	9.2	63.9	63.5	93.8	83.5
AddSub	46.7	70.6	67.8	74.4	82.0	83.8
SingleEq	43.2	74.0	75.9	73.6	79.3	84.9
GSM-8K	8.6	8.7	22.7	39.8	46.6	41.8
CSQA	41.8	63.7	57.1	66.9	64.7	80.2
StrategyQA	57.2	55.1	56.3	53.0	67.6	76.2
(Average)	37.00	46.88	57.28	61.87	72.33	75.05

Table 3: Baseline results on PaLM-62B (**Left**) and PaLM-540B (**Right**) (Chowdhery et al., 2022) without using self-consistency (i.e. argmax sampling). Except for not using self-consistency and the use of the zero temperature, all experimental setup is otherwise identical to that in Table 1.



(a) PaLM-62B



(b) PaLM-540B

Figure 7: Distribution of entropy (Eq. (6)) in Stage 1 (i.e. first m sampling paths with Zero-shot CoT without demonstrations) and in Stage 2 (i.e. the second m sampling paths after demonstrations are prepended) in PaLM models.

Model	**PaLM-62B**	
Setting	**0-shot**	
Method	COSP (*Ours*)	COSP^{+} (*Ours*)
# Paths	7+7	7+7
MultiArith	85.0	**86.2**
AddSub	78.9	**79.2**
SingleEq	78.7	**78.9**
GSM-8K	**30.2**	28.2
CSQA	60.2	**60.3**
StrategyQA	64.7	**65.1**
(Average)	66.28	**66.32**

Table 4: Comparison between COSP and COSP^{+} on PaLM-62B. While COSP uniformly uses 5 in-context demonstrations per question, COSP^{+} uses an adaptive number of demonstrations per sample proportional to the outcome entropy of the test question in Stage 1 (Eq. (6)) and uses a minimum of 5 and up to 8 demonstrations per sample. The COSP results are lifted from Table 1.

C.5 Failure Analysis

In this section we conduct analysis in cases where demonstrations do not lead to performance improvement or even leads to performance deterioration. The only case where COSP performs worse than the Zero-shot CoT is on the GSM-8K task on GPT-3. We show some of the identified demonstrations on this particular task-model combination in Table 7 in App. D, and we find that in this case, due to the extreme difficulty of the task to the model most of the generated demonstrations are wrong. On the other hand, competing methods like Auto-CoT lead to such performance deterioration more frequently. Intuitively, unlike COSP where we use self-consistency to approximate correctness, the decision of Auto-CoT on which question and rationale to include entirely hinges on the question embedding, and when the initial Zero-shot performance is low, most of the generated responses would be erroneous. We find that such deterioration indeed happens more frequently in the smaller models and/or more difficult tasks where the LLMs rely more on in-context examples for guidance (we give an example in Table 6. On the other hand, in larger models like PaLM-540B with strong Zero-shot performance and whose errors are more likely due to minor flaws in reasoning (e.g. arithmetic mistakes in a single step) rather than major issues (e.g. repetitions, gibberish outputs & major semantic misunderstanding), we find that consistent with the arguments in §5.5 of Zhang et al. (2022), the

Model	**GPT-3**		**PaLM-62B**	
Setting	**5-shot**			
Method	Manual CoT	COSP-FS	5-shot CoT	COSP-FS
# Paths	14	7+7	14	7+7
GSM-8K	16.7	**17.7**	30.3	**30.8**

Table 5: COSP-FS results.

results are indeed less sensitive to flawed demonstrations.

D Examples of Generated Demonstrations

We show some of the generated demonstrations in various datasets in Table 7 (GPT-3) and Table 8 (PaLM).

COSP	Q: Irwin 's family went on a camping trip in the mountains . On the first day , they hiked from their car to the campsite . First , they hiked 0.2 mile from the car to a stream , and 0.4 mile from the stream to a meadow . Then they hiked 0.1 mile from the meadow to the campsite . How many miles did Irwin 's family hike in all ? A: Let's think step by step. First, they hiked 0.2 miles from their car to the stream. Then, they hiked 0.4 miles from the stream to the meadow. Finally, they hiked 0.1 miles from the meadow to the campsite. So, the total distance is 0.2 + 0.4 + 0.1 = 0.7 miles. Therefore, the answer (arabic numerals) is 0.7. ✓ Q: Joan purchased a basketball game for $ 5.20 , and a racing game for $ 4.23 . How much did Joan spend on video games ? A: Let's think step by step. 1. Joan purchased a basketball game for $ 5.20 , and a racing game for $ 4.23 . 2. $ 5.20 + $ 4.23 = $ 9.43 3. $ 9.43 is the total amount spent on video games. Therefore, the answer (arabic numerals) is 9.43. ✓ Q: Jason has 7 violet balloons and 4 red balloons . He lost 3 of the violet balloons . How many violet balloons does Jason have now ? A: Let's think step by step. 7 - 3 = 4 4 = 4 So, he has 4 violet balloons now. Therefore, the answer (arabic numerals) is 4. ✓ Q: Joan decided to sell all of her old books . She gathered up 33 books to sell . She sold 26 books in a yard sale . How many books does Joan now have ? A: Let's think step by step. 1. 33 books - 26 books = 7 books 2. Therefore, the answer (arabic numerals) is 7. ✓ Q: Keith grew 6 turnips . Alyssa grew 9 turnips . How many turnips did they grow in all ? A: Let's think step by step. 1. Keith grew 6 turnips. 2. Alyssa grew 9 turnips. 3. So in total, they grew 15 turnips. Therefore, the answer (arabic numerals) is 15. ✓
Auto-CoT	Q: Sally picked 7 lemons and Mary picked 9 lemons from the lemon tree . How many lemons were picked in total ? A: Let's think step by step. Sally picked 7 lemons. Mary picked 9 lemons. So, 14 lemons were picked in total. Therefore, the answer (arabic numerals) is 14. × Q: Sally had 27 Pokemon cards . Dan gave her 41 new Pokemon cards . Sally bought 20 Pokemon cards . How many Pokemon cards does Sally have now ? A: Let's think step by step. Sally had 27 Pokemon cards. Dan gave her 41 new Pokemon cards. Sally bought 20 Pokemon cards. How many Pokemon cards does Sally have now? Therefore, the answer (arabic numerals) is 27. × Q: Mary had 21 dimes and 38 pennies in her bank . Her dad borrowed 18 pennies from Mary . How many pennies does she have now ? A: Let's think step by step. Step 1: 21 dimes + 38 pennies = 59 pennies Step 2: 18 pennies - 18 pennies = 0 pennies Step 3: 59 - 0 = 59 pennies Step 4: 59 pennies = 59 Step 5: 59 pennies = 59 pennies Step 6: 59 pennies = 59 Step 7: 59 = 59 Step 8: Therefore, Mary has 59 pennies. Therefore, the answer (arabic numerals) is 59. × Q: There are 33 oak trees currently in the park . Park workers had to cut down 18 oak trees that were damaged . How many oak trees will be in the park when the workers are finished ? A: Let's think step by step. - First, we know that 18 trees were cut down. - 18 trees were cut down, so there were originally 15 trees. - 15 - 18 = -3 trees, so there will be 3 new oak trees in the park. - 33 - 3 = 30 trees, so there will be 30 oak trees in the park when the workers are finished. Therefore, the answer (arabic numerals) is 30. × Q: When Jake had 1 cat , he needed to serve 0.5 can of cat food each day . Now that Jake has adopted a second cat , he needs to serve a total of 0.9 can each day . How much extra food is needed to feed the second cat ? A: Let's think step by step. - First, we need to find the total number of cans needed to feed both cats. - Then, we need to find the difference between the total number of cans and the number of cans needed to feed one cat. - Finally, we need to find the number of cans needed to feed the second cat. - The answer is the number of cans needed to feed the second cat. Therefore, the answer (arabic numerals) is 0.9. ×

Table 6: Comparison between the demonstrations identified by COSP and Auto-CoT on the AddSub task using GPT-3. In this case, the COSP demonstrations lead to 18% improvement over Zero-shot CoT with self-consistency whereas AutoCoT demonstration led to performance deterioration. Reasoning steps are highlighted: correct, questionable/incomplete/repetitive/irrelevant and erroneous, and the ground-truth correctness of the demonstrations (unavailable to the LLM) marked by ✓ or × at the end of the responses.

Dataset	Demonstrations
Multi Arith	Q: Debby's class is going on a field trip to the zoo. If each van can hold 4 people and there are 2 students and 6 adults going, how many vans will they need? A: Let's think step by step. First, we'll figure out how many people are going. 2 students and 6 adults. So, 8 people. Then, we divide the number of people by the number of people in a van. 8 people divided by 4 people per van = 2 vans. Therefore, the answer (arabic numerals) is 2. ✓
	Q: Nancy had 80 files on her computer. She deleted 31 of them and put the rest into folders with 7 files in each one. How many folders did Nancy end up with? A: Let's think step by step. She deleted 31 files, so she had 80 - 31 = 49 files. She put 7 files in each folder, so she had 49 / 7 = 7 folders. Therefore, the answer (arabic numerals) is 7. ✓
	Q: At the arcade, Tom won 32 tickets playing 'whack a mole' and 25 tickets playing 'skee ball'. If he spent 7 of his tickets on a hat, how many tickets does Tom have left? A: Let's think step by step. He has 32 + 25 = 57 tickets. He spends 7 on a hat, so he has 50 left. Therefore, the answer (arabic numerals) is 50 ✓
	Q: The schools debate team had 5 boys and 40 girls on it. If they were split into groups of 9 how many groups could they make? A: Let's think step by step. 5 boys and 40 girls = 45 students Let's take 9 students per group. Therefore, the answer (arabic numerals) is 5. ✓
SingleEq	Q: Mike has 87 baseball cards. Sam bought 13 of Mike's baseball cards. How many baseball cards does Mike have now ? A: Let's think step by step. 13 of Mike's baseball cards are with Sam. So, 87 - 13 = 74. Then, Mike has 74 baseball cards now. Therefore, the answer (arabic numerals) is 74.✓
	Q: Sam had 98 pennies in his bank. He spent 93 of his pennies. How many pennies does he have now ? A: Let's think step by step. 1. Sam had 98 pennies in his bank. 2. He spent 93 pennies, so the amount of pennies he has left is 5. 3. So, Sam has 5 pennies now. """ # Solution # Sam had 98 pennies in his bank. # He spent 93 pennies, so the amount of pennies he has left is 5. # So, Sam has 5 pennies now. # 2. Therefore, the answer (arabic numerals) is 5. ✓
	Q: Nancy goes fishing with Joan. They catch 18 trout. If they equally split up the trout, how many will each one get ? A: Let's think step by step. First, we divide the total number of trout by the number of people. The answer is 9. So, each person catches 9 trout. Therefore, the answer (arabic numerals) is 9. ✓
	Q: Mrs. Hilt baked pies last weekend for a holiday dinner. She baked 16 pecan pies and 14 apples pies. If she wants to arrange all of the pies in rows of 5 pies each, how many rows will she have? A: Let's think step by step. 16 pecan pies + 14 apples pies = 30 pies. 30 pies / 5 pies per row = 6 rows. So, 6 rows. Therefore, the answer (arabic numerals) is 6. ✓
GSM-8K	Q: Two thirds of Jana's puppies are Pomeranians. One third of the Pomeranians are girls. If there are 6 Pomeranian girls, how many puppies does Jana have? A: Let's think step by step. First, we'll look at the full set of options for Jana's puppies: Puppies: Jana's puppes are Pomeranians and are either girls or boys. Pomeranians: Jana's puppies are Pomeranians. Girls: Jana's puppies are girls. Boys: Jana's puppies are boys. Let's look at the information we have, and see what we know for sure. Jana has 6 puppies. Jana's puppies are all Pomeranians. Therefore, the answer (arabic numerals) is 6. ×
	Q: If one person can paint half a house in 5 days, how many hours would it take for 5 people to paint a whole house? A: Let's think step by step. 1. Half a house can be painted in 5 days by 1 person. 2. 5 people can paint half a house in 5 days. 3. 5 people can paint a whole house in 5 days. So the answer is 5 days. Therefore, the answer (arabic numerals) is 5. ×
	Q: The local firefighters are doing a "fill the boot" fundraiser. Their goal is to raise \$6300. After the first 3 hours, they have raised \$2100. For how many hours do they have to fundraise in total to reach their goal, assuming an equal amount raised in every hour? A: Let's think step by step. We know that the total amount raised is equal to the total number of hours multiplied by the amount raised per hour. Total amount raised = Total number of hours * amount raised per hour \$6300 = 3 * \$2100 \$6300 / \$2100 = 3 So, the total number of hours is 3. Therefore, the answer (arabic numerals) is 3. ×
	Q: A glass of milk is 8 ounces of milk. John drinks 2 glasses of milk. If milk has 3 calories per ounce how many calories did he consume? A: Let's think step by step. John drank 2 glasses of milk. Each glass had 8 ounces. Each ounce had 3 calories. So, John consumed 2 x 8 x 3 = 48 calories. Therefore, the answer (arabic numerals) is 48. ✓

CSQA	Q: What are our bodies doing after having food? Answer Choices: (A) falling down (B) digesting (C) gas (D) weight gain (E) not hungry A: Let's think step by step. After having food, our bodies are digesting it. Digestion is the process by which food is broken down into simpler substances. Therefore, among A through E, the answer is B. ✓ Q: The wood was still rough to the touch, what did the woodworker have to do? Answer Choices: (A) petrify (B) sanded (C) warp (D) composted (E) clean A: Let's think step by step. First, the woodworker had to petrify the rough wood. Then, the woodworker had to sand the petrified wood. Finally, the woodworker had to warp the sanded wood. Therefore, among A through E, the answer is B. ✓ Q: What type of store would have lots of sports equipment? Answer Choices: (A) mall (B) office supply store (C) school (D) sporting goods store (E) sporting event A: Let's think step by step. I'm a sports fan. I would go to sporting events, not to malls. Therefore, among A through E, the answer is D. ✓ Q: What do people aim to do at work? Answer Choices: (A) complete job (B) learn from each other (C) kill animals (D) wear hats (E) talk to each other A: Let's think step by step. People aim to do their jobs. Therefore, among A through E, the answer is A. ✓
Strategy QA	Q: Has the Indian Ocean garbage patch not completed two full rotations of debris since its discovery? A: Let's think step by step. The Indian Ocean garbage patch is approximately 10% of the size of the North Pacific garbage patch. This means that even if the Indian Ocean garbage patch has completed two full rotations of debris, it will still only be 1/10th of the size of the North Pacific garbage patch. Therefore, the answer (Yes or No) is no. ✗ Q: Was Christina Aguilera born in the forgotten borough? A: Let's think step by step. 1. Christina 2. Aguilera 3. was 4. born 5. in 6. the 7. forgotten 8. Therefore, the answer (Yes or No) is yes. ✗ Q: Karachi was a part of Alexander the Great's success? A: Let's think step by step. 1. Alexander the Great had a huge army. 2. Karachi was a part of the empire of Alexander the Great. 3. Therefore, Alexander the Great had a huge army. 4. Therefore, Karachi was a part of Alexander the Great's success. Therefore, the answer (Yes or No) is no. ✓ Q: Was Jean Valjean imprisoned due to hunger? A: Let's think step by step. yes 1. Jean Valjean was imprisoned for stealing bread. 2. He was imprisoned for stealing bread because he was hungry. 3. Therefore, the answer (Yes or No) is yes. Therefore, the answer (Yes or No) is yes. ✓

Table 7: Annotated examples of generated demonstrations selected by COSP on **GPT-3.** Similar to Table 6, reasoning steps are highlighted: correct, questionable/incomplete/repetitive/irrelevant and erroneous, and the ground-truth correctness of the demonstrations (unavailable to the LLM) marked by ✓ or ✗ at the end of the responses.

Dataset	Demonstrations
GSM-8K	Q: Bruce works for 5 hours on Tuesday. On Wednesday he works twice the time he works on Tuesday. On Thursday he works 2 hours less than the time he works on Wednesday. How many hours does Bruce work in all these three days? A: Let's think step by step. On Tuesday, Bruce works for 5 hours. On Wednesday, Bruce works for 2 * 5 = 10 hours. On Thursday, Bruce works for 10 - 2 = 8 hours. So, in all three days, Bruce works for 5 + 10 + 8 = 23 hours. Therefore, the answer (arabic numerals) is 23. ✓ Q: Mark decides to buy packs of old magic cards and open them to sell. He buys 3 packs for $1800 each. He gets 1 card that is worth $4000 and another card worth $1000. There are 30 more cards worth an average of $50 each. How much money profit did he make? A: Let's think step by step. He spent $1800 * 3 = $5400. He got $4000 + $1000 + $50 * 30 = $5500. So he made $100 profit. Therefore, the answer (arabic numerals) is 100. ✓
CSQA	Q: Where would someone purchase an upright piano? Answer Choices: (A) music class (B) college (C) music store (D) music room (E) music band A: Let's think step by step. You can't buy a piano in a music class, college, or room. And an upright piano is not in a music band. The correct answer is a music store. Therefore, among A through E, the answer is C. ✓ Q: John got his tax refund back. He treated it like it was what? Answer Choices: (A) candy (B) death and (C) free money (D) discount (E) credit A: Let's think step by step. The first thing John did when he got his tax refund was to spend it on his girlfriend. The second thing he did was to spend it on a big-screen TV. The correct answer is C. Therefore, among A through E, the answer is C. ✓ Q: How are the conditions for someone who is living in a homeless shelter? Answer Choices: (A) sometimes bad (B) happy (C) respiration (D) growing older (E) death A: Let's think step by step. The answer is choice A because the passage describes that homeless people have poor health. Therefore, among A through E, the answer is A. ✓

Table 8: Annotated examples of generated demonstrations selected by COSP on **PaLM-540B.** See explanations of the annotations in Table 7