Self-AMPLIFY : Improving Small Language Models with Self Post Hoc Explanations

Milan Bhan[1,2], Jean-Noël Vittaut[1], Nicolas Chesneau[2] and Marie-Jeanne Lesot[1]

[1]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[2]Ekimetrics, Paris, France
{milan.bhan, nicolas.chesneau}@ekimetrics.com
{jean-noel.vittaut, marie-jeanne.lesot}@lip6.fr

Abstract

Incorporating natural language rationales in the prompt and In-Context Learning (ICL) have led to a significant improvement of Large Language Models (LLMs) performance. However, generating high-quality rationales require human-annotation or the use of auxiliary proxy models. In this work, we propose `Self-AMPLIFY` to automatically generate rationales from post hoc explanation methods applied to Small Language Models (SLMs) to improve their own performance. `Self-AMPLIFY` is a 3-step method that targets samples, generates rationales and builds a final prompt to leverage ICL. `Self-AMPLIFY` performance is evaluated on four SLMs and five datasets requiring strong reasoning abilities. `Self-AMPLIFY` achieves good results against competitors, leading to strong accuracy improvement. `Self-AMPLIFY` is the first method to apply post hoc explanation methods to autoregressive language models to generate rationales to improve their own performance in a fully automated manner.

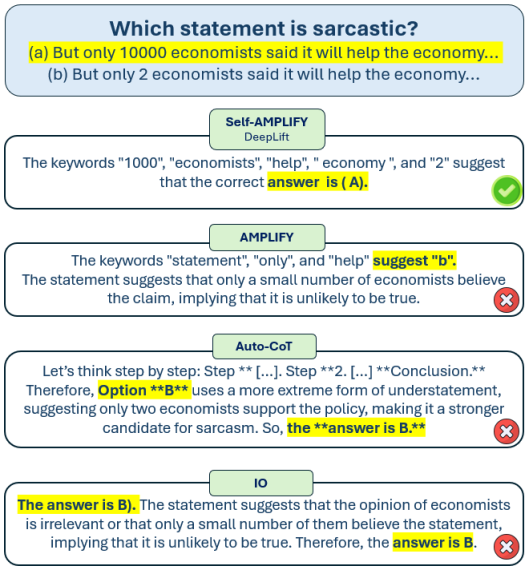

Figure 1: Example of four responses to a question from the Snarks dataset, generated from different ICL prompting strategies. Traditional input-output (`IO`) prompting, `Auto-CoT` (Zhang et al., 2023) and `AMPLIFY` (Krishna et al., 2023) fail to answer properly, whereas `Self-AMPLIFY` generates important tokens as a rationale before correctly answering.

1 Introduction

Autoregressive Large Language Models (LLMs) such as GPT-3 (Brown et al., 2020), PaLM (Chowdhery et al., 2023) or LaMDA (Thoppilan et al., 2022), have made significant advancements in a wide range of NLP tasks. These models have demonstrated so-called "emergent abilities" (Schaeffer et al., 2024), including in-context learning (ICL), instruction following and reasoning (Wei et al., 2022). ICL (see Dong et al. (2023) for a recent survey) involves learning from a few examples integrated into the prompt without fine tuning the model.

LLMs' emergent abilities have been leveraged to enhance performance by incorporating human-annotated intermediate reasoning steps within the context, called *rationales* (Wei et al., 2024). By learning to sequentially generate (1) the reasoning steps through rationales and (2) the final answer, LLMs have reached state-of-the-art performance in complex tasks requiring reasoning abilities such as commonsense or symbolic reasoning. To overcome the need for human annotation, automatic rationale generation methods have been proposed. `AMPLIFY` (Krishna et al., 2023) has demonstrated that rationales can be generated from smaller proxy supervised Language Models (LMs) to enrich the prompt to enhance the performance of LLMs. `AMPLIFY` targets promising instances to be integrated into the final prompt using the proxy model and automatically builds rationales based on post hoc attribution explanation methods (Molnar, 2020) applied to this proxy model.

Recently, small autoregressive LMs (SLMs),

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 10974–10991
November 12-16, 2024

with fewer than 14 billion parameters, have emerged, such as Mistral (Jiang et al., 2023), Zephyr (Tunstall et al., 2023) or Gemma (Gemma Team, 2024). They achieve performance sometimes approaching that of LLMs' on common benchmarks: their smaller size makes them computationally efficient while maintaining a high level of accuracy. In particular, classical post hoc attribution methods such as `KernelSHAP` (Lundberg and Lee, 2017) or `DeepLift` (Shrikumar et al., 2017) become affordable to explain SLMs' prediction, despite their high computational cost of these methods.

In this paper, we propose `Self-AMPLIFY`, an extension of the `AMPLIFY` framework for SLMs that does not need an auxiliary model nor human annotations. The main contributions of the `Self-AMPLIFY` framework are as follows: (i) promising instances to be integrated into the final prompt are targeted only using the considered SLM's prediction, (ii) post hoc explanation methods are applied to the SLM itself to automatically generate rationales as a self-improving signal, (iii) three types of post hoc explanations methods are implemented: post hoc attributions, self `topk` explanations and self free text rationales.

As an illustration, Figure 1 shows three responses to a question from the Snarks (Srivastava et al., 2023) dataset respectively obtained using the proposed `Self-AMPLIFY`, a classical prompting approach, `IO`, a rationale enhanced approach, `Auto-CoT` (Zhang et al., 2023) and `AMPLIFY`. `Self-AMPLIFY` succeeds to generate the good answer whereas its three competitors fail.

Experimental results discussed in Section 4 show that `Self-AMPLIFY` leads to a performance gain on a wide range of datasets as compared to `IO`, `Auto-CoT` and `AMPLIFY`. As a result, we show that post hoc explanation methods of various kinds can be directly applied to the SLM to generate automatically rationales to self-improve. Unlike the original `AMPLIFY` framework, proxy fine tuned models are no longer needed to increase LMs' performance, making `Self-AMPLIFY` more autonomous and flexible.

2 Background and Related Work

In this work, we consider in-context learning (ICL), where a few samples are provided to an autoregressive LM within the prompt to perform a particular NLP task. In this section we recall some basic principles of post hoc explanations and existing methods that generate rationales to enhance LMs' performance by enriching prompts.

2.1 Post Hoc Explanations Background

Attribution method. Attribution methods compute an importance score for each input feature to explain the model output. Two types of methods can be distinguished: *perturbation-based* and *gradient-based* (Zhao et al., 2024).

The former perturbs and resamples feature values to compute feature importance. Two common examples are LIME (Ribeiro et al., 2016) and KernelSHAP (Lundberg and Lee, 2017). However, these methods are computationally expensive due to the numerous inferences required.

On the other hand, gradient-based approaches estimate feature importance through the model backpropagated gradient activity. Two common examples are Integrated Gradients (Sundararajan et al., 2017) and DeepLift (Shrikumar et al., 2017). However, these methods are computationally expensive due to the need to compute gradients. Therefore, to the best of our knowledge, they have not been yet applied to autoregressive LLMs.

Post hoc free text self-rationales. Free text rationales are natural language intermediate reasoning steps that justify a model's prediction (see Gurrapu et al. (2023) for a recent survey) or favor reasoning in LLMs (Huang and Chang, 2023). Post hoc self-rationale generation involves directly prompting LM's to explain their prediction in free text given their answer (Huang et al., 2023; Madsen et al., 2024). Post-hoc self-rationales contrast with attribution numerical vector explanations in terms of their higher level of abstraction.

2.2 Related Work

This section introduces two categories of methods for generating rationales aimed at enriching the prompt and encouraging LLMs to engage in reasoning rather than merely providing answers.

Human-annotated rationales. Firstly, rationales can be generated manually. Several handcrafted benchmarks have been proposed to either train language models to generate rationales or to assess language models' ability to generate rationales aligned with human annotations, such as e-SNLI (Camburu et al., 2018) or

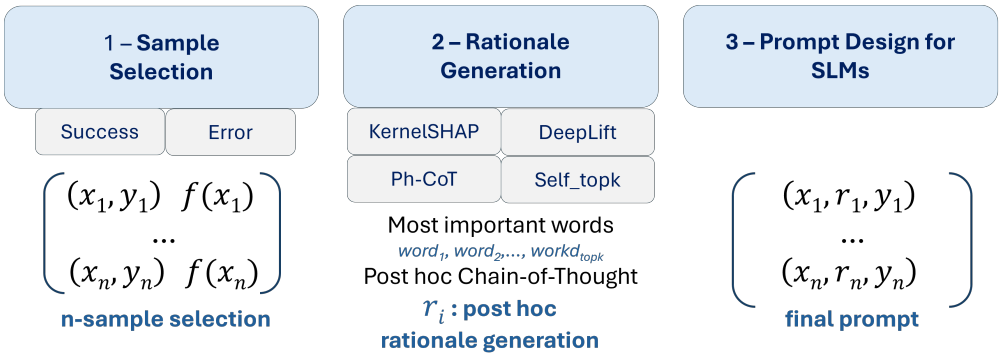

Figure 2: `Self-AMPLIFY` overview. `Self-AMPLIFY` is a 3-step approach generating rationales to self-improve a SLM in a ICL setting. (1) Promising samples are targeted following two selection strategies: `success` or `error`. (2) Rationales are generated based on a post hoc explanation method: `KernelShap`, `DeepLift`, `Ph-CoT` or `Self_topk`. (3) The final ICL prompt is built based on the previously generated rationales.

ERASER (DeYoung et al., 2020). Chain-of-Thought (CoT) (Wei et al., 2024) adds human-annotated rationale steps to the standard ICL prompt template (x, y) to construct a *explain-then-predict* template (x, r, y) where x is the input text, y is the expected answer and r is the provided rationale. CoT extensions have been proposed to aggregate multiple reasoning paths (Wang et al., 2023) or to enable LLMs to explore multiple promising reasoning paths (Yao et al., 2024) during text generation. These approaches significantly improve LLMs' performance on NLP tasks requiring reasoning capabilities. Another way of using rationales to enrich the ICL prompt consists in appending the rationale after the answer in a *predict-then-explain* manner, as (x, y, r), resulting in a relative performance gain (Lampinen et al., 2022) as compared to the (x, r, y) design.

However relying on human-annotated rationales makes these methods costly and not automatable. Moreover, they require strong reasoning capabilities and often yield significant performance gains only for LLMs larger than a certain size (Wei et al., 2024).

Automatically generated rationales. Automatic rationale generation eliminates the need for human-annotated rationales. Automatic Chain-of-Thought prompting (`Auto-CoT`) (Zhang et al., 2023) proposes to generate automatically natural language rationales by prompting the LLM to "think step by step". A Sentence-Transformer (Reimers and Gurevych, 2019) is used to cluster input texts in order to generate one CoT rationale per cluster, making the approach dependent on this auxiliary Sentence Transformer. Then, the LLM's prediction $\hat{y}$ is integrated to construct the final prompt $(x, r, \hat{y})$. However, `Auto-CoT` is prone to include incorrect demonstrations and low-quality samples in the prompt, since it does not take the ground truth answer for the final prompt construction.

`AMPLIFY` (Krishna et al., 2023) automatically generates rationales from post hoc numeric attribution methods from an auxiliary fine tuned proxy model. The latter is initially fine tuned on a corpus of interest to generate relevant explanations. Then, a n-shot sample selection is performed using the same proxy model to identify misclassified instances. These samples are then added to the ICL prompt, following a (x, r, y) template. Therefore, `AMPLIFY` relies heavily on the use of the auxiliary proxy model, both at the n-shot targeting and the rationale generation steps. While `AMPLIFY` yields significant performance gain as compared to classical prompting, it has only been tested on GPT-3 and GPT-3.5. Moreover, `AMPLIFY` does not incorporate free text rationales in its framework.

3 Proposed approach: `Self-AMPLIFY`

This section describes the architecture of `Self-AMPLIFY`, an extension of the `AMPLIFY` (Krishna et al., 2023) framework. As sketched in Figure 2 and detailed in the next subsections, this framework enriches prompts with self-generated rationales in a fully automated manner to enhance SLMs' performance in ICL settings. By generating rationales directly from the SLM, `Self-AMPLIFY` differs from `AMPLIFY` in that it does not depend on any auxiliary fine-tuned proxy model and the data used to train

it. Therefore, post-hoc explanation methods are leveraged to self-improve SLM fully automatically.

3.1 Self-AMPLIFY overview

As shown in Figure 2 and detailed in the following, Self-AMPLIFY is a 3-step approach that takes as input an autoregressive SLM f and a corpus of texts $\mathcal{T}$ from which the n-shot sample is generated. Each input text is associated with an expected answer, belonging to a label space denoted $\mathcal{L}$. The code is available online on a public repository[1].

(i) n-shot Sample Selection. This step aims to select input texts that will be added to the final prompt. Self-AMPLIFY employs two simple yet efficient selecting strategies only based solely on f prediction, eliminating the need of an auxiliary model as in the AMPLIFY framework.

(ii) Rationale Generation. Rationales are generated for the previously selected texts by applying post hoc explanation methods to f itself. This way, unlike AMPLIFY, rationales are not generated from a fine tuned side proxy model. We implements 3 types of post-hoc explanation methods to generate rationales directly from f, making Self-AMPLIFY more versatile.

(iii) Prompt Design for SLMs. The final prompt is constructed based on the previously generated rationales. Each generated rationale is added between its related input text and ground truth answer. The enriched sample is finally used to make the prediction on the test set.

3.2 n-shot Sample Selection

The first step involves selecting n instances from the text corpus $\mathcal{T}$ for inclusion in the final prompt.

Self-AMPLIFY employs two selection strategies based solely on f prediction: success and error. The success strategy selects text instances correctly predicted by f in a standard prompt setting, whereas the error strategy selects ones incorrectly predicted. To determine if an instance of interest $x \in \mathcal{T}$ is correctly predicted, we append the text "The answer is" to the initial prompt to guide f next token prediction. Therefore, the next token is more likely to be predicted in the correct format as in Kojima et al. (2022), i.e with the next token predicted in the label space $\mathcal{L}$. Denoting y the ground truth, the model prediction is categorized as a success if $f(x) = y$ and an error if $f(x) \neq y$ with $f(x) \in \mathcal{L}$. Otherwise, x is disgarded.

The success strategy relies on the idea that "the higher the prediction certainty, the more relevant the explanation" (Bhan et al., 2023a). Conversely, the error strategy relies on the idea that adding misclassified examples may avoid similar misclassifications on the test set. We assess the impact of the selection strategy on f performance in Section 4. This way, regardless of the selection strategy, Self-AMPLIFY does not rely on a proxy additional model to select samples, making it more flexible than other methods.

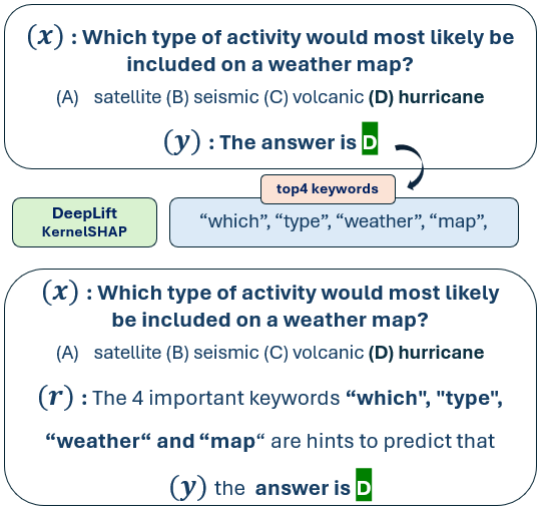

Figure 3: Self-AMPLIFY rationale generation step with a post hoc attribution method. Here, DeepLift or KernelShap is applied to the SLM to explain the answer D. The 4 most important tokens are targeted and the final rationale r is constructed based on these keywords. The (x, r, y) triplet is finally added to the ICL prompt.

3.3 Rationale Generation

The rationale generation step is summarized in Figure 3. Once the n-shot sample is created, rationales are generated by computing post hoc explanation from f directly. Self-AMPLIFY differs from AMPLIFY in that it generates rationales without the use of an auxiliary fine tuned model. In addition, Self-AMPLIFY implements 3 types of post hoc explanations to generate natural language rationale: post hoc attributions (DeepLift and KernelSHAP), post hoc Self_topk explanations and post hoc CoT (Ph-CoT) rationales where AMPLIFY only implements attribution methods, making Self-AMPLIFY more versatile. Post-hoc explanations are computed to explain each (x, y)

[1] https://github.com/milanbhan/self_amplify

pair to finally build their associated rationales r.

DeepLift and KernelShap are computed to explain the (x, y) pair, i.e. f output neuron related to y. DeepLift decomposes the neural network prediction by backpropagating the contributions of all neurons in the network to each input feature. Attribution scores are computed with respect to a chosen baseline. We define this baseline so that attribution is only computed on the input text, disregarding the special tokens or instruction text in the prompt. KernelSHAP samples instances in the neighborhood of x to approximate Shapley Values. In the same way as DeepLift, we only perturb input tokens belonging to the input text, disregarding the rest of the prompt. Therefore, attribution is only computed on tokens from the instance of interest. Appendix A.5 provides more details about post hoc attribution implementation.

The k tokens with the highest attribution score are then selected to build the rationale: it is defined following the template *"The k keywords $\langle word_1 \rangle$, $\langle word_2 \rangle$,..., and $\langle word_k \rangle$ are important to predict that the answer is $\langle y \rangle$"*. This way, Self-AMPLIFY generates rationales from post hoc attribution methods by converting a numerical vector of importance into a natural language rationale.

Self_topk consists in directly prompting f to generate the k most important tokens used to make its prediction. Self_topk is generated in a *predict-then-explain* post hoc manner, since the text containing the k most important keywords is generated given the ground truth answer y.

Finally, Ph-CoT consists in prompting f to generate a p-step free text explanation in a post hoc manner, given the ground truth y. Therefore, Ph-CoT can be defined as a post hoc Chain-of-Thought explanation. The final related rationale r is defined following the template *"p-step rationale: $\langle \phi \rangle$, therefore the answer is $\langle y \rangle$"*, where ϕ is the post-hoc free text rationale previously generated, and p is the number of steps in the rationale. Appendix A.4 provides more details about the prompts used to generate Self_topk and Ph-CoT rationales. We give several examples of generated rationales and answers conditioned by different rationale generator in Appendix A.7.

3.4 Prompt Design for SLMs

The final step consists in designing the prompt that is used to make the prediction on the test set.

We define a preprompt at the beginning of the final prompt to define the instruction asked to f, i.e. generating a rationale and an answer to a specific question. The preprompt can take two different forms, depending on the format of the generated rationales (top_k important words or p-step natural language explanation). More details about the preprompt are provided in Appendix A.4.

Finally, the output prompt is built based on the previously generated rationales. The latter is built following the template: *"*preprompt*, (x_1, r_1, y_1), (x_2, r_2, y_2), ..., (x_n, r_n, y_n)"*. Finally, this n-shot prompt is used as a context to make predictions in an ICL setting on the test set.

4 Experimental Settings

This section presents the experimental study conducted across five datasets and three autoregressive SLMs of various size. We start by running two versions of Self-AMPLIFY on two 7 billion parameters, respectively based on two post hoc explainers (DeepLift and Ph-CoT). We compare these two versions of Self-AMPLIFY to Auto-CoT and AMPLIFY, two competitors automatically generating rationales and IO (Input-Output), a traditional prompting setup baseline. Next, we deeply assess the impact of the *topk* post hoc explainers on Self-AMPLIFY performance through an ablation study. Finally, we run Self-AMPLIFY and the Gemma SLM in its 2 and 7 billion parameter versions and highlight the limits of our approach on tiny models.

4.1 Experimental protocol.

Datasets. Self-AMPLIFY is tested on five common LMs' benchmarks. ARC Challenge (Clark et al., 2018), CommonsenseQA (CQA) (Talmor et al., 2019) and Social IQa (SIQA) (Sap et al., 2019) are commonsense reasoning datasets requiring the ability to use prior knowledge about the world. The Snarks and Causal Judgment datasets (Srivastava et al., 2023) are datasets related to challenging complex tasks. Snarks requires to distinguish between sarcastic and non-sarcastic sentences, and Causal Judgment is designed to assess the ability in deducing causal factors from a detailed summary. These datasets are commonly used to evaluate LMs' performance.

Models. We test Self-AMPLIFY on Instruction-tuned SLMs whose size does not exceed 7 billion parameters and achieve good results in common benchmarks. Mistral-7B (Jiang

Model (size)	Dataset	Selection strategy	IO (ref.)	Auto-CoT	AMPLIFY	Self-AMPLIFY (ours)	
					BERT proxy	DeepLift	Ph-CoT
Mistral (7B)	ARC Challenge	Success	72.8	71.8	70.4	71.1	**75.2***
		Error	69.0	69.3	70.4	70.0	**72.8***
	Causal Judgment	Success	36.8	**63.2*****	52.6***	52.6***	50.0***
		Error	31.6	50.0**	39.5	55.3***	**60.5*****
	CQA	Success	60.7	61.3	60.7	66.7**	**67.6*****
		Error	61.7	59.3	64.7	62.3	**66.3***
	SIQA	Success	57.3	60.0	56.0	59.7	**62.7****
		Error	59.3	55.3	62.7*	61.7	**63.0***
	Snarks	Success	50.0	**66.7***	55.6	58.3	63.9*
		Error	36.1	50.0*	47.2	52.8**	**72.2*****
Zephyr (7B)	ARC Challenge	Success	63.6	63.3	67.0*	66.0	**70.7*****
		Error	65.3	65.6	**71.1****	68.4*	68.0
	Causal Judgment	Success	39.5	55.3**	50.0	52.6*	**57.9****
		Error	42.1	50.0	**60.5****	47.3	52.6*
	CQA	Success	53.3	61.0***	61.3***	**64.7*****	62.3***
		Error	56.3	63.0**	**68.0*****	63.3**	66.7***
	SIQA	Success	53.7	59.7**	56.7	59.0**	**65.0*****
		Error	51.0	60.0***	59.3***	**60.3*****	54.3*
	Snarks	Success	36.1	**44.4***	**44.4***	41.7	38.9
		Error	47.2	41.7	41.7	52.8	**55.6***

Table 1: Self-AMPLIFY and competitors accuracy (%) on five test sets and two 7 billion parameters models. Self-AMPLIFY is tested on 2 versions, depending on the post hoc explainer used to generate rationales. IO stands for "input-output" standard prompting. Auto-CoT and AMPLIFY are two competing methods automatically generating rationales to enhance the input prompt. The best results are highlighted in bold. With p as the p-value of the one-tailed paired t-test, $*p < 10\%$, $**p < 5\%$, $***p < 1\%$. IO (ref.) stands for the reference baseline.

et al., 2023), Zephyr-7B (Tunstall et al., 2023) and Gemma-7B (Gemma Team, 2024) are 7 billion parameters SLMs achieving state-of-the-art performance among other SLMs in a wide variety of NLP tasks. We then test the limits of Self-AMPLIFY on the smaller 2 billion parameter SLM Gemma-2b achieving strong performance for its size but with less reasoning abilities.

Self-AMPLIFY versions and competitors. We test four instantiations of the Self-AMPLIFY framework based on the four following post hoc explanation methods: DeepLift, KernelShap, Self_topk and Ph-CoT. In particular, we compared the two DeepLift and Ph-CoT instanciations of Self-AMPLIFY to a traditional (x, y) prompting setup (input-output, IO), Auto-CoT (Zhang et al., 2023) and AMPLIFY (Krishna et al., 2023). For a fair comparison, we run Self-AMPLIFY, Auto-CoT and AMPLIFY with the same n-shot sample context. This way, we focus our comparative analysis on the ability of each method to generate high quality rationales leading to accuracy improvement. AMPLIFY rationales are generated by applying DeepLift to a fine tuned BERT-base model. We give more details about the AMPLIFY implementation in Appendix A.2.

Self-AMPLIFY and its competitors are tested on the same corpora. Therefore, contexts are enriched from the same training corpus $\mathcal{T}$ and inference is performed on the same test sets. Finally, the output is retrieved in the correct format by using the assessed SLM as in Zhang et al. (2023) to fit the label space (for example A or B for Snarks) and to compute accuracy. Because of the high computational cost of testing, we vary the number of runs and the size of test sets according to the performed analysis. We detail sample sizes, number of runs, context size (n), number of keywords (k) and number of steps (p) associated with each SLM and dataset in our experiments in Appendix A.3. We present an in-depth analysis of the impact of n and k on Self-AMPLIFY performance in Appendix A.6.

4.2 Results.

Global Results. Table 1 shows the experimental results obtained on Mistral-7B and Zephyr-7B by running once Self-AMPLIFY with DeepLift and Ph-CoT and its competitors on the same train set for rationale generation and the same test set for performance assessment. For each dataset/model case, one of the Self-AMPLIFY modalities leads to

Dataset	Selection strategy	Self-AMPLIFY			
		KernelShap	DeepLift	Self_topk	random
ARC Challenge	Success	**69.0 ± 2.3**	67.7 ± 2.0	67.9 ± 2.7	64.9 ± 5.0
	Error	67.6 ± 2.7	67.7 ± 2.9	**69.3 ± 1.6**	63.7 ± 4.4
Causal Judgment	Success	49.2 ± 5.0	**49.7 ± 4.9**	48.2 ± 5.6	45.8 ± 4.1
	Error	47.4 ± 5.1	**52.6 ± 5.0**	**52.6 ± 6.1**	47.3 ± 6.0
CQA	Success	58.8 ± 2.1	58.9 ± 1.8	**59.9 ± 2.8**	58.4 ± 2.6
	Error	58.4 ± 2.9	59.7 ± 3.1	**61.1 ± 2.8**	57.2 ± 4.1
Snarks	Success	**50.6 ± 7.9**	50.0 ± 4.7	49.7 ± 6.1	50.0 ± 6.1
	Error	**53.6 ± 8.0**	52.0 ± 5.1	50.0 ± 5.4	47.8 ± 5.4
SIQA	Success	**59.6 ± 3.0**	59.0 ± 4.1	58.6 ± 2.5	54.9 ± 2.9
	Error	**60.6 ± 3.1**	60.0 ± 2.7	58.7 ± 4.0	55.9 ± 2.8
Average		*57,5*	*57,7*	*57,6*	*54,6*

Table 2: Average accuracy (%) and standard deviation computed on 10 Self-AMPLIFY runs for different *topk* post hoc explainers on Mistral-7B.

the best result (for example Ph-CoT related to the success strategy for the Mistral/ARC Challenge case). The two Self-AMPLIFY modalities perform almost always better than the classical IO prompting and lead in average to higher accuracy as compared to Auto-CoT. The two instanciations of Self-AMPLIFY perform better than AMPLIFY with Mistral-7B without the use of a proxy additional fine-tuned model to generate rationales. In particular, post hoc Ph-CoT rationales give in average the best Self-AMPLIFY results. These results confirm the interest of Self-AMPLIFY to improve SLMs' performance fully automatically.

Impact of the selection strategy. Table 1 highlights that the success selection strategy of Self-AMPLIFY gives good results overall, doing on average as well as the error one. This result confirms the interest of adding initially correctly classified examples into the context, which is not possible in the initial AMPLIFY framework. Self-AMPLIFY gives almost always better results than AMPLIFY when applied with the success strategy. However, AMPLIFY and Self-AMPLIFY have similar overall results with the error strategy. The results do not show whether a given selection strategy gives better results with Self-AMPLIFY.

Impact of the post hoc explainer. Table 1 shows that Ph-CoT post hoc explanations give in average the best Self-AMPLIFY results as compared to DeepLift. In particular, Ph-CoT related to the error strategy leads to significant highest performance gain compared to other competitors for complex tasks such as Snarks and Causal Judgment. We hypothesize that this is linked to the ability of SLMs to generate faithful free text natural language post hoc explanations as a corrective signal. Table 2 shows the results of the ablation study of the *topk* explanations on Mistral-7B obtained on 10 Self-AMPLIFY runs. The random *topk* rationale generator gives the worst accuracy compare to the other Self-AMPLIFY instantiations. This result shows that the rationale signal content can have an impact on the Self-AMPLIFY performance. The different *topk* instantiations of Self-AMPLIFY give very similar results on average, indicating that the framework is robust.

Based on these results, the default setting of Self-AMPLIFY *topk* explainer only depends on whether or not the model parameters are accessible. If DeepLift is much less computationally costly than KernelShap, it can only be applied if the model's internal parameters are accessible, which is not always the case with online APIs. Self_topk is less costly than DeepLift in that it is only based on text generation, without any additional computation. However, it is difficult to completely control the format of the *topk* explanations, as text generation does not always respect the template initially provided.

The size limit of post hoc rationale enhancement. We extend our analysis with two new SLMs: Gemma-7B and the tiny Gemma-2B. Figure 4 shows that on average, Self-AMPLIFY outperforms its competitors on Gemma-7B in the same way as with Zephyr-7B and Mistral-7B. Every version of Self-AMPLIFY consistently outperforms IO and AMPLIFY and do better on average than Auto-CoT. However, rationale enhancement leads to much poorer results with Gemma-2B as compared to Gemma-7B: Self-AMPLIFY, Auto-CoT and AMPLIFY do barely as well on average than the

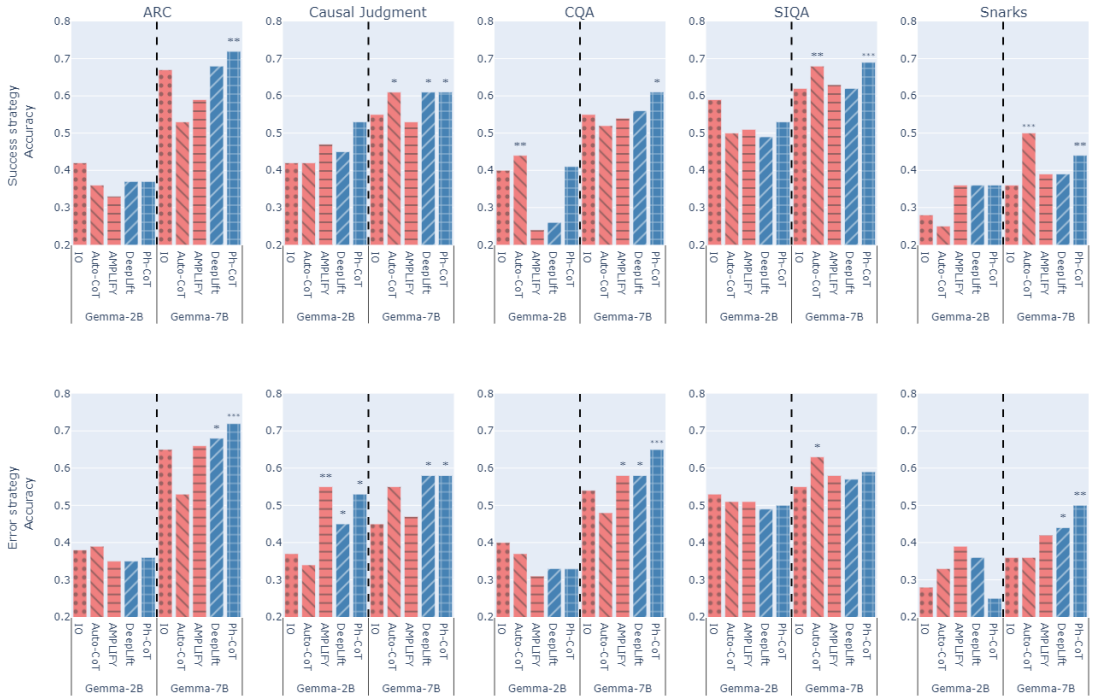

Figure 4: `Self-AMPLIFY` (blue) and competitors (red) accuracy (%) with Gemma-2 (left) and Gemma-7B (right). `Self-AMPLIFY` is run on 2 versions: `DeepLift` and `Ph-CoT`. With p as the p-value of the paired t-test, *$p < 10\%$, **$p < 5\%$, ***$p < 1\%$. `IO` stands for the reference baseline.

`IO` baseline. We hypothesize that these contrasting results are linked to `Gemma-2B`'s small size and less advanced reasoning capabilities as compared to 7-billion parameters models. `Gemma`'s results are presented with those obtained with `Zephyr-7B` and `Mistral-7B` in Appendix A.3 in Table 6.

5 Discussion

The `Self-AMPLIFY` framework is versatile and can work with any other post hoc attribution methods, such as Integrated Gradient or LIME. We recommend as `Self-AMPLIFY` default setting `Ph-CoT` rationales if the aim is only to obtain the most accurate results. However, a framework user might expect the generated rationales to be faithful to build trust with `Self-AMPLIFY` (Ferrario and Loi, 2022). Free text rationale faithfulness evaluation is a difficult task and there is no consensus in the way to measure it (Wiegreffe et al., 2021; Atanasova et al., 2023). Faithfulness assessment is easier with *topk* rationales by computing common metrics such as stability (Dai et al., 2022) and self-consistency (Madsen et al., 2024). Therefore, `KernelShap`, `DeepLift` or other post hoc attribution explainer should be preferred if rationale faithfulness evaluation is needed. The appropriate *topk* explainer can then be chosen depending on the level of information available about the model as stated in the previous section.

As a future work, `Self-AMPLIFY` could be improved by generating other types of rationales to enrich the prompt such as counterfactual examples (see Bhan et al. (2023b) for a recent method). A deeper analysis of the link between task complexity, selection strategy and `Self-AMPLIFY` performance would also provide information on how to better generate valuable rationales. Finally, it would be enlightening to assess the faithfulness of `Self-AMPLIFY` generated rationales. For instance, ICL generated rationales could be compared to ground truth explanations obtained in a post hoc manner. We see these perspectives as promising paths towards a better understanding of LMs' ability to faithfully learn to self-explain.

6 Conclusion

We introduced `Self-AMPLIFY`, an extension of the `AMPLIFY` framework, automatically generating rationales to enrich the prompt in ICL settings for SLMs. `Self-AMPLIFY` is the first approach

enriching the prompt without human-annotated rationales or the use of auxiliary models, but only with the SLM itself. Self-AMPLIFY implements 2 selection strategies and 4 post hoc explanation methods, making it versatile and flexible. Self-AMPLIFY results in performance gain compared to its competitors in the considered tasks for 7 billion parameters models. Finally, this work sheds light on the interest of using post hoc explanations to enhance SLM's performance.

7 Limitations

Datasets and models. In this work we have tested Self-AMPLIFY by applying it on 5 datasets and 3 language models. The conclusions of our work would have more weight if other models were included in the study. Furthermore, it would be interesting to apply Self-AMPLIFY on slightly bigger SLMs with better reasoning abilities. This would make the framework even more useful to the community.

Rationale relevance. The quality of the generated rationales is not assessed, neither when enriching the prompt (rationale generation step), nor when generating the text (prediction on the test set). These rationales should be interpreted with caution, as they have been generated solely to enhance SLMs' performance. This phenomenon has been raised by (Zhang et al., 2023), where wrong demonstrations based on low quality rationales can still lead to performance gains.

Computational cost. The use of KernelShap and DeepLift is computationally costly. Even if it is affordable to use them with SLMs, the resource requirement is substantial. One could lower the number of samples used to compute KernelShap if needed (see Appendix A.5) to make it even more affordable. On the other hand, the Self_topk and Ph-CoT instances of Self-AMPLIFY are much less costly, as they are only based on prompting, making their inference computational cost comparable to Auto-CoT and more advantageous than AMPLIFY.

Ethics Statement

Since SLMs' training data can be biased, there is a risk of generating harmful text during inference. One using Self-AMPLIFY to generate rationales must be aware of these biases in order to stand back and analyze the produced texts. Finally, SLMs consume energy, potentially emitting greenhouse gases. They must be used with caution.

Acknowledgment

We thank Gabriel Olympie and Jean Lelong for insights and enlightening conversations.

References

Pepa Atanasova, Oana-Maria Camburu, Christina Lioma, Thomas Lukasiewicz, Jakob Grue Simonsen, and Isabelle Augenstein. 2023. Faithfulness Tests for Natural Language Explanations. In *Proc. of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 283–294, Toronto, Canada. Association for Computational Linguistics.

Milan Bhan, Nina Achache, Victor Legrand, Annabelle Blangero, and Nicolas Chesneau. 2023a. Evaluating self-attention interpretability through human-grounded experimental protocol. In *Proc. of the First World Conf. on Explainable Artificial Intelligence xAI*, pages 26–46.

Milan Bhan, Jean-Noël Vittaut, Nicolas Chesneau, and Marie-Jeanne Lesot. 2023b. Tigtec: Token importance guided text counterfactuals. In *Proc. of the European Conf. on Machine Learning ECML-PKDD*, page 496–512. Springer.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

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*.

Jessica Dai, Sohini Upadhyay, Ulrich Aivodji, Stephen H Bach, and Himabindu Lakkaraju. 2022. Fairness via explanation quality: Evaluating disparities in the quality of post hoc explanations. In *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*, pages 203–214.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Procs. of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A Survey on In-context Learning. ArXiv:2301.00234 [cs].

Andrea Ferrario and Michele Loi. 2022. How explainability contributes to trust in ai. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1457–1466.

Google DeepMind Gemma Team. 2024. Gemma: Open models based on gemini research and technology.

Sai Gurrapu, Ajay Kulkarni, Lifu Huang, Ismini Lourentzou, and Feras A. Batarseh. 2023. Rationalization for Explainable NLP: A Survey. *Frontiers in Artificial Intelligence*, 6.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards Reasoning in Large Language Models: A Survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065. Association for Computational Linguistics.

Shiyuan Huang, Siddarth Mamidanna, Shreedhar Jangam, Yilun Zhou, and Leilani H. Gilpin. 2023. Can Large Language Models Explain Themselves? A Study of LLM-Generated Self-Explanations. ArXiv:2310.11207 [cs].

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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, NeurIPS22*, 35:22199–22213.

Satyapriya Krishna, Jiaqi Ma, Dylan Slack, Asma Ghandeharioun, Sameer Singh, and Himabindu Lakkaraju. 2023. Post Hoc Explanations of Language Models Can Improve Language Models. In *Advances in Neural Information Processing Systems, NeurIPS23*.

Andrew Lampinen, Ishita Dasgupta, Stephanie Chan, Kory Mathewson, Mh Tessler, Antonia Creswell, James McClelland, Jane Wang, and Felix Hill. 2022. Can language models learn from explanations in context? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 537–563. Association for Computational Linguistics.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proc. of the 31st Int. Conf. on Neural Information Processing Systems*, NIPS'17, pages 4768–4777.

Andreas Madsen, Sarath Chandar, and Siva Reddy. 2024. Are self-explanations from large language models faithful. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Vivek Miglani, Aobo Yang, Aram Markosyan, Diego Garcia-Olano, and Narine Kokhlikyan. 2023. Using Captum to Explain Generative Language Models. In *Proc. of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 165–173. Association for Computational Linguistics.

Christoph Molnar. 2020. *Interpretable Machine Learning*. Lulu.com.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144. Association for Computing Machinery.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. 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 4463–4473,

Hong Kong, China. Association for Computational Linguistics.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2024. Are emergent abilities of large language models a mirage? *Advances in Neural Information Processing Systems*, 36.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *Proc. of the 34th Int. Conf. on Machine Learning, ICML*, volume 70 of *ICML'17*, pages 3145–3153. JMLR.org.

Aarohi Srivastava, Denis Kleyjo, and Ziyi Wu. 2023. Beyond the imitation game: Quantifying and extrapolatingthe capabilities of language models. *Transactions on Machine Learning Research*, (5).

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proc. of the 34th Int. Conf. on Machine Learning, ICML*, volume 70 of *ICML'17*, pages 3319–3328. JMLR.org.

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*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct Distillation of LM Alignment. ArXiv:2310.16944 [cs].

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *Proc. of the 11th Int. Conf. on Learning Representations, ICLR23*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Chain-of-thought prompting elicits reasoning in large language models. In *Proc. of the 36th Int. Conf. on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Sarah Wiegreffe, Ana Marasović, and Noah A Smith. 2021. Measuring association between labels and free-text rationales. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10266–10284.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proc. of the Conf. on Empirical Methods in Natural language Processing: system demonstrations, EMNLP*, pages 38–45.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic Chain of Thought Prompting in Large Language Models. In *Proc. of the 11th Int. Conf. on Learning Representations, ICLR23*.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for Large Language Models: A Survey. *ACM Transactions on Intelligent Systems and Technology*.

A Appendix

A.1 Scientific libraries

We used several open-source libraries in this work: pytorch (Paszke et al., 2019), HuggingFace transformers (Wolf et al., 2020) sklearn (Pedregosa et al., 2011) and Captum (Miglani et al., 2023).

A.2 SLMs implementation Details

Small Language Models. The library used to import the pretrained SLMs is HuggingFace. In particular, the backbone version of Mistral is `Mistral-7B-Instruct-v0.2` the one of Zephyr is `zephyr-7b-beta`, and the one of Gemma are respectively `gemma-1.1-7b-it` and `gemma-1.1-2b-it`.

Instruction special tokens. The special tokens to use SLMs in instruction mode were the followings:

- `Mistral-7B-Instruct-v0.2`
  - `user_token='[INST]'`

  - assistant_token='[/INST]'
  - stop_token='</s>'
- Zephyr-7b-beta
  - user_token='<|user|>'
  - assistant_token='<|assistant|>'
  - stop_token='</s>'
- Gemma-1.1-2b-it
  - user_token='<start_of_turn>user'
  - assistant_token='<start_of_turn>model'
  - stop_token='<eos>'
- Gemma-1.1-7b-it
  - user_token='<start_of_turn>user'
  - assistant_token='<start_of_turn>model'
  - stop_token='<eos>'

Text generation. Text generation was performed using the native functions of the Hugging Face library: `generate`. The `generate` function has been used with the following parameters:

- `max_new_tokens = 300`
- `do_sample = True`
- `num_beams = 2`
- `no_repeat_ngram_size = 2`
- `early_stopping = True`
- `temperature = 0.95`

AMPLIFY implementation AMPLIFY has been implemented by fine tuning one BERT-base per training set. Table 3 gives more information about AMPLIFY proxy models used to generate $topk$ explanations to enhance the prompt. The $topk$ post-hoc explanation method used was DeepLift.

A.3 Experimental protocol details

Table 4 shows the experimental protocol details of the performed analysis. Test set size is 39 for Snarks and 36 for Causal Judgment for every experiment. However, test sets are obtained by randomly sampling for ARC Challenge, CQA and SIQA with a varying size. Number of runs can also vary from one experiment to another. This is due to the high computational cost of running Self-AMPLIFY and its competitors with various selection strategy modalities on such a high number of datasets and texts. Since the ablation study only concerns 3 Self-AMPLIFY modalities and a random baseline, experiment contains 10 runs. Test size is however smaller for ARC Challenge, CQA and SIQA.

Table 5 presents the hyperparameters of Self-AMPLIFY and the context size of the experiments. Post hoc attribution methods and Self_topk are computed with $k = 6$ for Mistral-7B, Zephyr-7B and Gemma-7B and $k = 4$ for Gemma-2B. Ph-CoT p-step rationales are generated with $p = 3$. ICL context size is set at $n = 8$ for Zephyr-7B, Mistral-7B and Gemma-7B for all the datasets, except for Causal Judgment, where $n = 6$. The ICL size is set at $n = 4$ for Gemma-2B, this smaller model being less able to handle long contexts.

The results of the experimental protocol are all presented in Table 6.

Dataset	Accuracy (%)	nb epoch	Proxy model
ARC Challenge	25.7	25	BERT-base
Causal Judgment	52.6	25	BERT-base
CQA	20.9	25	BERT-base
Snarks	61.1	25	BERT-base
SIQA	56.8	10	BERT-base

Table 3: AMPLIFY proxy models performance and number of epochs by dataset.

		Experiment		
		Mistral-7B and Zephyr-7B	**Mistral-7B $topk$ ablation study**	**Gemma-7B and Gemma-2B**
Corresponding table/figure		Table 1	Table 2	Figure 4
Nb runs		1	10	1
Test set size	**ARC Challenge**	295	150	295
	Causal Jugment	36	36	36
	CQA	300	150	300
	Snarks	39	39	39
	SIQA	300	150	300

Table 4: Experimental protocols details. Number of runs and test set sizes vary depending on the performed analysis.

A.4 Prompting format

Here we provide some details of different prompts used to give instructions to SLMs.

Prompt for Self_topk rationale generation

Hyperparameter	Mistral-7B, Zephyr-7B, Gemma-7B	Gemma-2B
Context size n	6 (Causal Judgment) 8 (other datasets)	4
Number of keywords in topk rationales k	6	4
Number of steps in Ph-CoT p	3	3

Table 5: `Self-AMPLIFY` hyperparameters per model per dataset.

user
Choose the right answer with the ⟨topk⟩ *most important keywords used to answer. Example: The answer is (A), the* ⟨topk⟩ *most important keywords to make the prediction are "word$_1$", ... and "word$_k$"*

Preprompt for `Ph-CoT` rationale generation
user
Choose the right answer and generate a concise ⟨n_steps⟩*-step explanation, with only one sentence per step. Example: The answer is (A),* ⟨n_steps⟩*-step explanation: step$_1$, step$_2$,...,step$_n$.*

Final ICL n-samples prompt example based on *topk* rationales
user
You are presented with multiple choice question, where choices will look like (A), (B), (C) or (D), generate ⟨topk_words⟩ *keywords providing hints and generate the right single answer Ouput example: The* ⟨topk_words⟩ *keywords "word$_1$", "word$_2$" ... and "word$_k$" are important to predict that the answer is (A)*
⟨ question$_1$ ⟩
assistant
⟨ rationale$_1$ ⟩
⟨ answer$_1$ ⟩
...
user
⟨ question$_n$ ⟩
assistant
⟨ rationale$_n$ ⟩
⟨ answer$_n$ ⟩
user
⟨ question$_{n+1}$⟩

A.5 Post hoc attribution explanation methods

Captum library. Post hoc attribution has been computed using the Captum (Miglani et al., 2023) library. `Self-AMPLIFY` implements additional post hoc attribution methods as compared to those presented in our paper. These additional post hoc attribution methods can be used in the `Self-AMPLIFY` framework to generate rationales. Overall, we implement the following methods:

- Gradient-based
  - `GradientXActivation`
  - `IntegratedGradients`
  - `DeepLift`
- Perturbation-based
  - `FeatureAblation`
  - `Lime`
  - `KernelShap`
  - `ShapleyValueSampling`
  - `ShapleyValues`

Attribution implementation details. In particular, gradient-based approach are computed with respect to the SLM embedding layer (`layer = model.model.embed_tokens`).

The parameters used to computed `DeepLift` and `KernelShap` were Captum's default settings. In particular, `KernelShap` was computed with `n_samples = 350`

Baseline choice. The baseline choice is decisive for DeepLift computation. The baseline is selected so that importance is only computed with respect to the initial prompt, so that special tokens and preprompt have a null attribution. The baseline is thus constructed as a modified version of the text on which DeepLift is applied. Therefore, the part of the baseline where the attribution must have a non-zero value (here statement, question and possible answer) is replaced with padding.

A.6 Impact of *topk* explanation length and context size

Figure 5 shows the evolution of the accuracy of `Self-AMPLIFY` with respect to the *topk* hyperparameter. It turns out that the *topk* explanation length does not seem to have an impact on the accuracy. Every *topk* value gives better results than `IO prompting`. Figure 6 shows the evolution of the accuracy of `Self-AMPLIFY` and `IO` prompting with respect to context size. Evaluation is made with `Mistral` and `Zephyr` and the Causal Judgment dataset. Most context sizes result in better `Self-AMPLIFY` result as compared to `IO`.

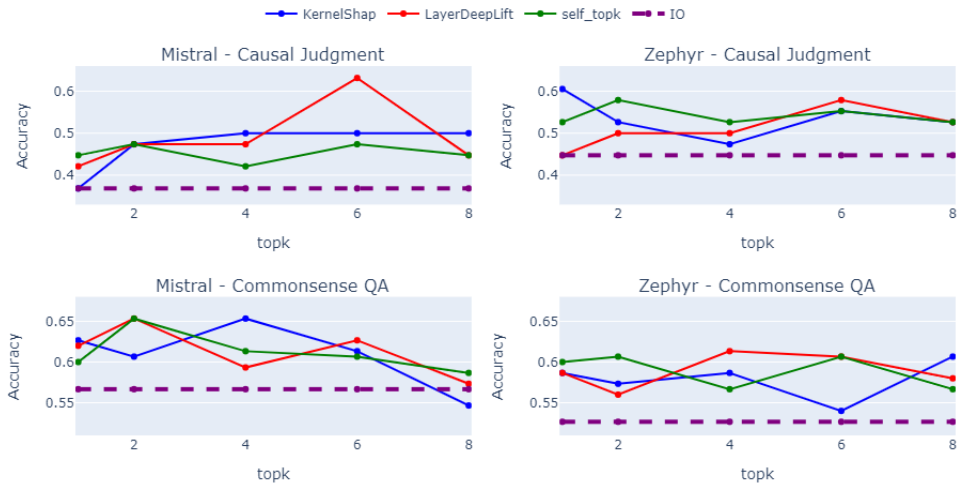

Figure 5: Accuracy (%) of classical IO prompting and `Self-AMPLIFY` for different *topk* post hoc explainers and different `topk` values. Evaluation is made with `Mistral` and `Zephyr` on Commonsense QA and Causal Judgment datasets.

A.7 `Self-AMPLIFY` and competitors generated text example

Figure 7, 8, 9, 10 and 11 show several examples of generated texts conditioned by different rationale generators for every analyzed datasets.

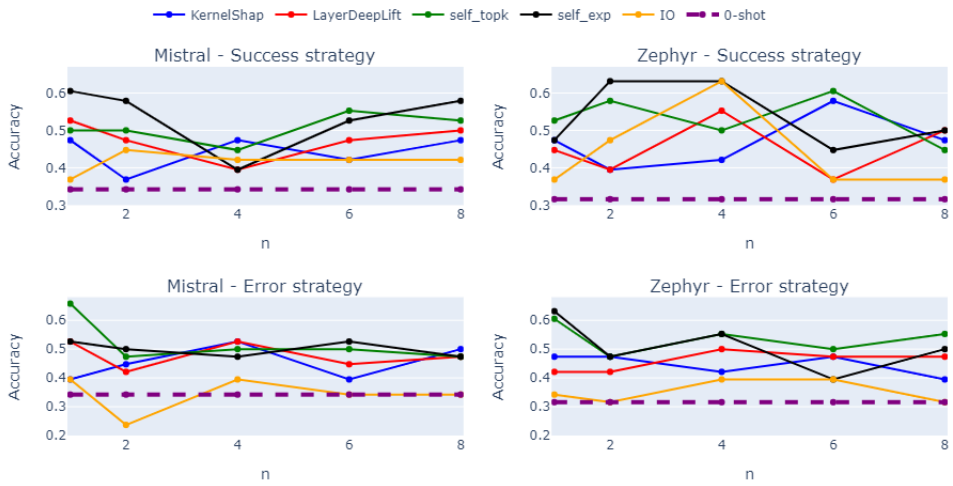

Figure 6: Accuracy (%) of `Self-AMPLIFY` and classical IO prompting for different context sizes. Evaluation is made with `Mistral` and `Zephyr` for every selection strategy on the Causal Judgment dataset.

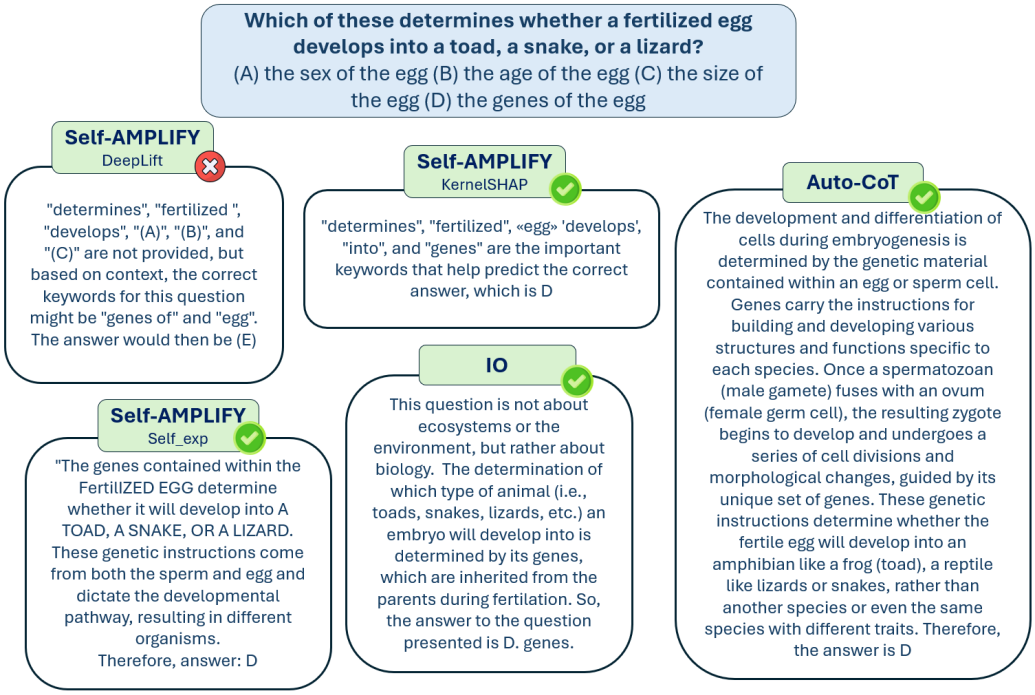

Figure 7: ARC Challenge answers conditioned by different ICL prompt built from different rationale generators.

Model (size)	Dataset	Selection strategy	IO (ref.)	Auto-CoT	AMPLIFY BERT proxy	Self-AMPLIFY (ours) DeepLift	Ph-CoT
Mistral (7B)	ARC Challenge	Success	72.8	71.8	70.4	71.1	**75.2***
		Error	69.0	69.3	70.4	70.0	**72.8***
	Causal Judgment	Success	36.8	**63.2*****	52.6***	52.6***	50.0***
		Error	31.6	50.0**	39.5	55.3***	**60.5*****
	CQA	Success	60.7	61.3	60.7	66.7**	**67.6*****
		Error	61.7	59.3	64.7	62.3	**66.3***
	SIQA	Success	57.3	60.0	56.0	59.7	**62.7****
		Error	59.3	55.3	62.7*	61.7	**63.0***
	Snarks	Success	50.0	**66.7***	55.6	58.3	63.9*
		Error	36.1	50.0*	47.2	52.8**	**72.2*****
Zephyr (7B)	ARC Challenge	Success	63.6	63.3	67.0	66.0	**70.7*****
		Error	65.3	65.6	**71.1****	68.4*	68.0
	Causal Judgment	Success	39.5	55.3**	50.0	52.6	**57.9****
		Error	42.1	50.0	**60.5****	47.3	52.6*
	CQA	Success	53.3	61.0***	61.3***	**64.7*****	62.3***
		Error	56.3	63.0**	**68.0*****	63.3**	66.7***
	SIQA	Success	53.7	59.7**	56.7	59.0**	**65.0*****
		Error	51.0	60.0***	59.3***	**60.3*****	54.3*
	Snarks	Success	36.1	**44.4***	**44.4***	41.7	38.9
		Error	47.2	41.7	41.7	52.8	**55.6***
Gemma (7B)	ARC Challenge	Success	66,7	52,7	59,2	68,0	**71,8****
		Error	64,6	52,7	65,6	67,7*	**71,8*****
	Causal Judgment	Success	55,3	**60,5***	52,6	**60,5***	**60,5***
		Error	44,7	55,3	47,4	**57,9***	**57,9***
	CQA	Success	54,7	51,7	53,7	56,3	**61,0***
		Error	54,0	48,3	57,7*	57,7*	**65,0*****
	SIQA	Success	61,7	67,7**	63,3	62,0	**68,7*****
		Error	55,3	**62,7****	58,0	56,7	58,7
	Snarks	Success	36,1	**50,0*****	38,9	38,9	44,4**
		Error	36,1	36,1	41,7	44,4**	**50,0****
Gemma (2B)	ARC Challenge	Success	**41,8**	36,1	32,7	37,4	37,1
		Error	38,4	**38,8**	34,7	34,7	36,1
	Causal Judgment	Success	42,1	42,1	47,4	44,7	**52,6***
		Error	36,8	34,2	**55,3****	44,7*	52,6**
	CQA	Success	39,7	**44,3****	23,7	26,0	41,3
		Error	**39,7**	37,0	31,3	33,0	32,7
	SIQA	Success	**59,3**	50,0	50,7	49,0	53,0
		Error	**52,7**	50,7	51,3	49,3	49,7
	Snarks	Success	27,8	25,0	**36,1**	**36,1**	**36,1**
		Error	27,8	33,3	**38,9**	36,1	25,0

Table 6: Self-AMPLIFY and competitors accuracy (%) on five test sets, on Mistral-7B, Zephyr-7B, Gemma-7B and Gemma-2B. Self-AMPLIFY is tested on 2 versions, depending on the post hoc explainer used to generate rationales. IO stands for "input-output" standard prompting. Auto-CoT and AMPLIFY are two competing methods automatically generating rationales to enhance the input prompt. The best results are highlighted in bold. With p as the p-value of the one-tailed paired t-test, $*p < 10\%$, $**p < 5\%$, $***p < 1\%$. IO (ref.) stands for the reference baseline.

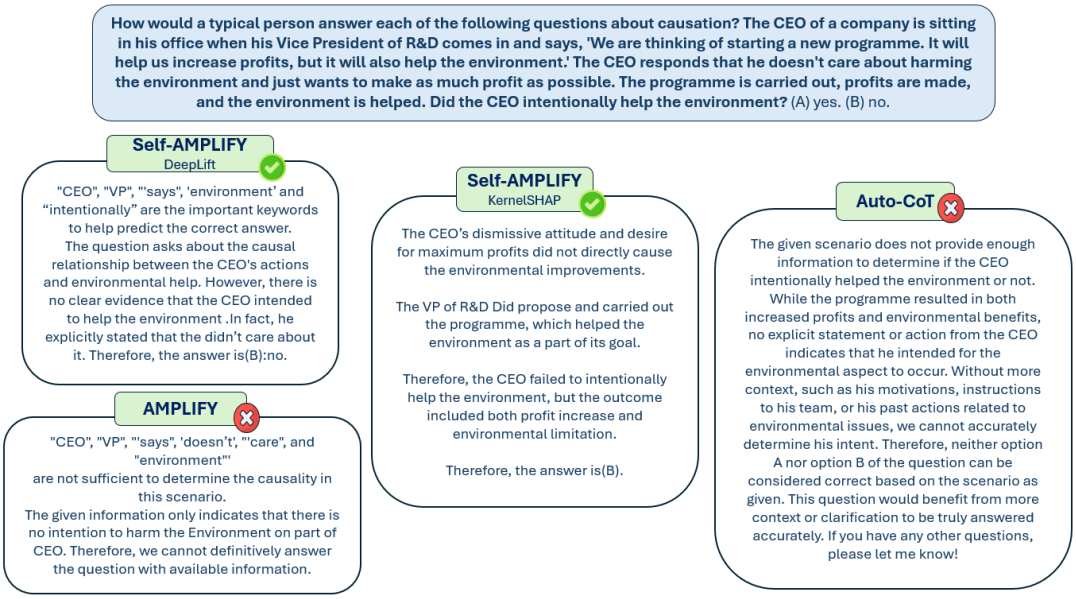

Figure 8: Causal Judgment answers conditioned by different ICL prompt built from different rationale generators.

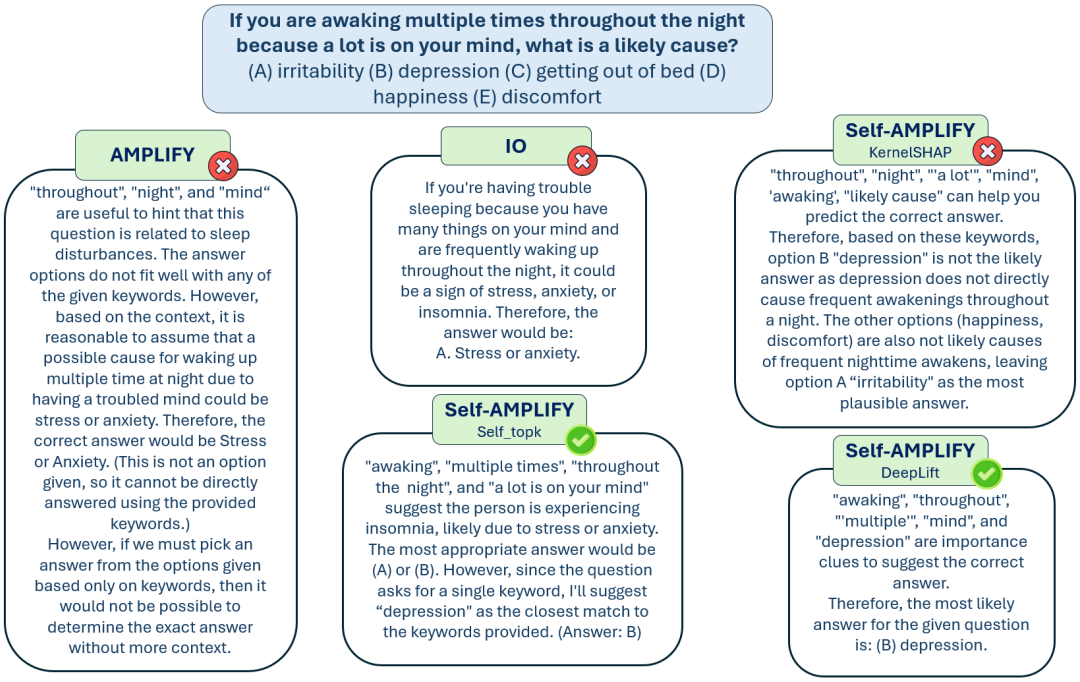

Figure 9: Commonsense QA answers conditioned by different ICL prompt built from different rationale generators.

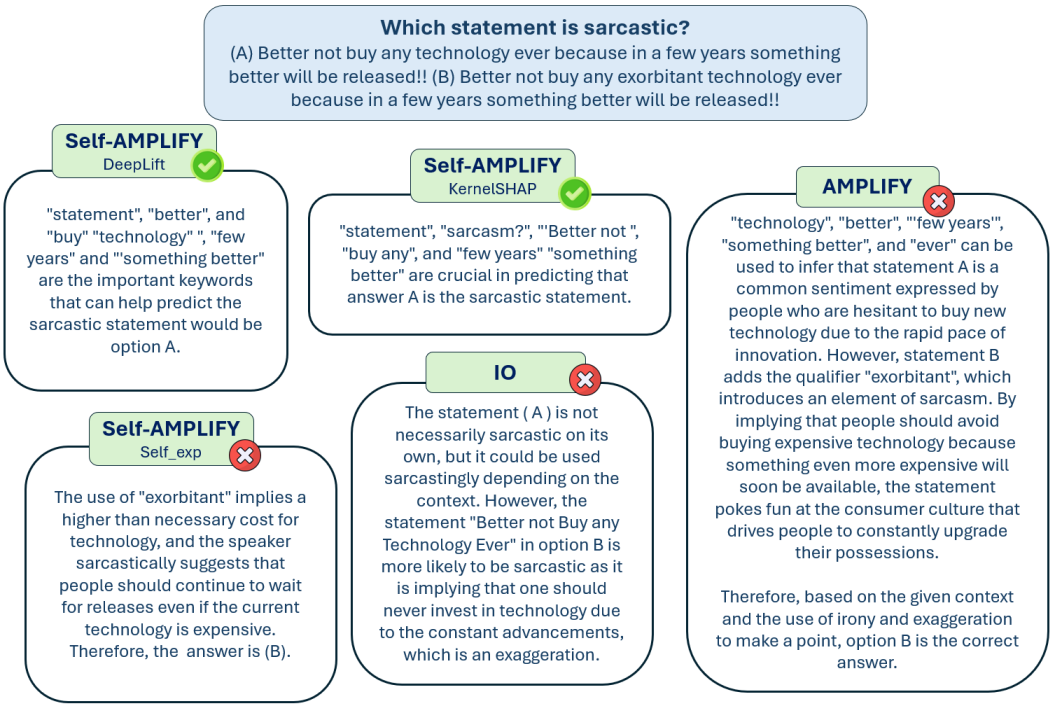

Figure 10: Snarks answers conditioned by different ICL prompt built from different rationale generators.

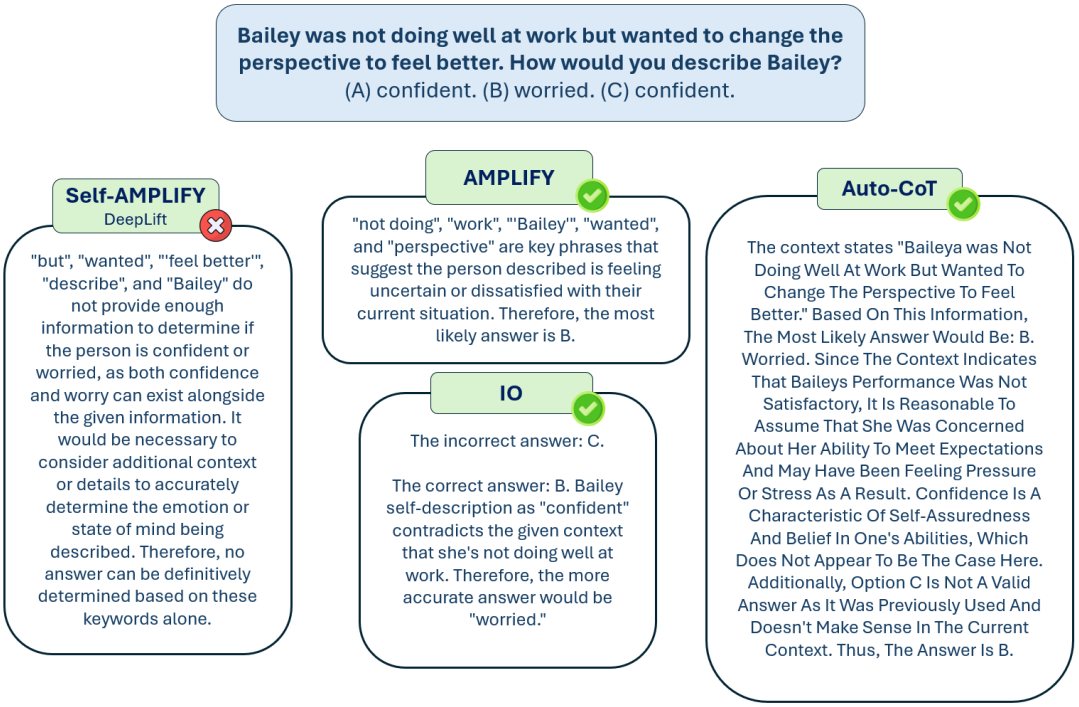

Figure 11: SIQA answers conditioned by different ICL prompt built from different rationale generators.