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

Anonymous ACL submission

Abstract

Incorporating natural language rationales in the prompt and In-Context Learning (ICL) has led to a significant improvement of Large Language Models (LLMs) performance. However, rationales currently require human-annotation or the use of auxiliary proxy models to target promising samples or generate high-quality rationales. In this work, we propose `Self-AMPLIFY` to generate automatically 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 two SLMs and two datasets requiring reasoning abilities: these experiments show that `Self-AMPLIFY` achieves good results against competitors. `Self-AMPLIFY` is the first method to apply post hoc explanation methods to SLM to generate rationales to improve their own performance in a fully automated manner.

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., 2023), 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., 2023). 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),

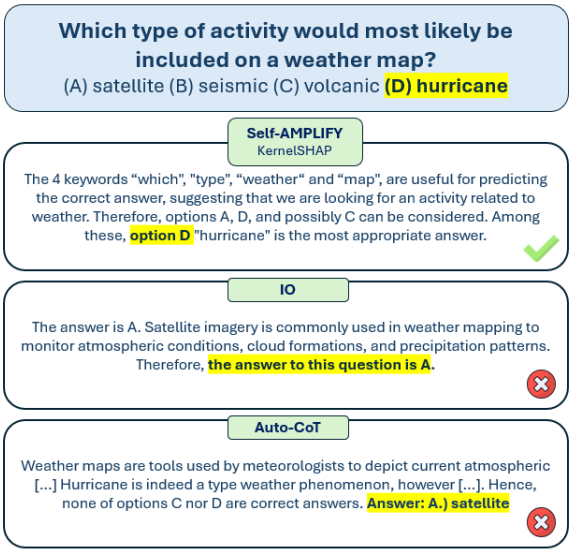

Figure 1: Example of three responses to a question from the ARC Challenge dataset, generated from different ICL prompting strategies. Traditional input-output (IO) prompting and `auto-CoT` (Zhang et al., 2023) fail to generate the good answer, whereas `Self-AMPLIFY` is able to generate important tokens as a rationale before answering as expected.

with fewer than 14 billions parameters, have emerged, such as Mistral (Jiang et al., 2023), Zephyr (Tunstall et al., 2023) or Llama-2-7B (Touvron et al., 2023). 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 `Self-AMPLIFY` 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 generate automatically rationales as a self-improving signal, (iii) three types of post hoc explanations methods are implemented: post hoc attributions, self `topk` explanations and self natural language explanations.

As an illustration, Figure 1 shows three responses to a question from the ARC Challenge (Clark et al., 2018) dataset respectively obtained using the proposed `Self-AMPLIFY`, a classical prompting approach, `IO`, and a rationale enhanced approach, `Auto-CoT` (Zhang et al., 2023). `IO` and `auto-CoT` fail to generate the good answer, while `Self-AMPLIFY` succeeds.

Experimental results discussed in Section 4 show that `Self-AMPLIFY` leads to a performance gain on sarcasm detection and commonsense reasoning as compared to `IO` and `auto-CoT`. 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 prompt.

2.1 Post Hoc Explanations Background

We recall two ways of generating post hoc explanations, respectively expressed using numerical vectors and natural language rationales.

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 Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al., 2016) and SHAPley additive explanations (SHAP) (Lundberg and Lee, 2017). However, these methods are computationally expensive due to the large number of 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 self-rationales. 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 natural language 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 lower 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. Chain-of-Thought

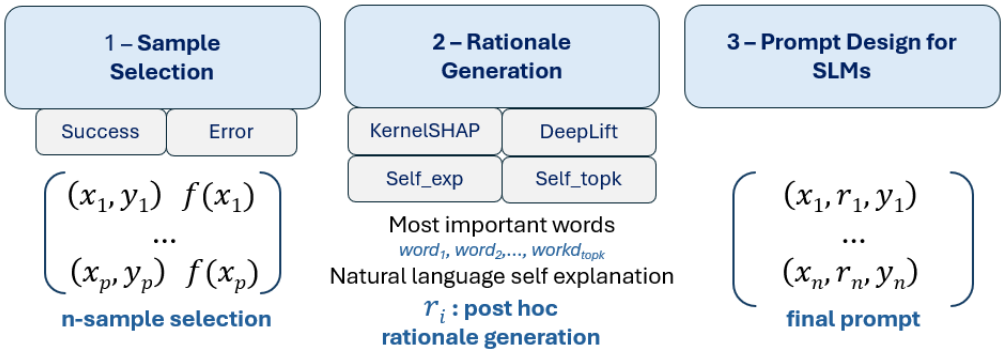

Figure 2: `Self-AMPLIFY` overview. `Self-AMPLIFY` is a 3-step approach generating rationales to self-improve a SLM in a ICL setting. (1) Promising samples are targeted following two selection strategies: `success` or `error`. (2) Rationales are generated based on a post hoc explanation method: `KernelShap`,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

(CoT) (Wei et al., 2023) adds human-annotated rationale steps in the prompt. The standard ICL template $(x,\ y)$ is enhanced to $(x,\ r,\ y)$ 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., 2023) 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, 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 non entirely automatable. Moreover, they require strong reasoning capabilities and often produce significant performance gains only for LLMs larger than a certain size (Wei et al., 2023).

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 with Zero-Shot-CoT (Kojima et al., 2022) 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 the $(x,\ r,\ y)$ 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 natural language rationales in its framework.

3 Proposed approach: `Self-AMPLIFY`

This section describes the architecture of `Self-AMPLIFY`, an extension of `AMPLIFY` for SLMs. 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 proxy model. Therefore, post-hoc explanation methods

are leveraged to self-improve SLM in a fully autonomous way.

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 $\mathcal{L}$. This framework is an extension of AMPLIFY (Krishna et al., 2023).

(i) n-shot Sample Selection. This step aims at selecting input texts that will be added to the final prompt. Self-AMPLIFY implements two simple yet efficient selecting strategies only based on f prediction, without the need of an auxiliary model.

(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 directly rationales from f, making Self-AMPLIFY more versatile.

(iii) Prompt Design for SLMs. The final prompt is built based on the rationales previously generated. 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 consists in selecting n texts from the corpus of interest $\mathcal{T}$ to be included in the final prompt.

Self-AMPLIFY implements 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 text instances incorrectly predicted by f. 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 a next token predicted in the label space $\mathcal{L}$. Denoting y the ground truth answer, the model prediction is considered as a success if $f(x) = y$, an error if $f(x) \neq y$ but $f(x)$ is still contained in the label space, it is discarded otherwise.

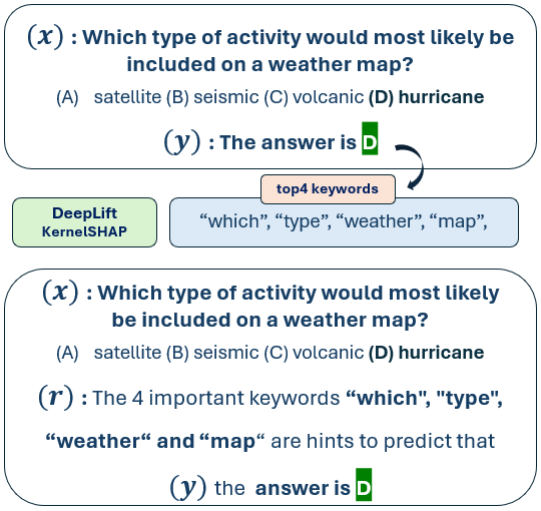

Figure 3: Self-AMPLIFY rationale generation step with a post hoc attribution method. Such a method (here DeepLift or KernelShap) is applied to the SLM to explain the answer D. The 4 most important tokens are targeted, and the final rationale r is generated based on the 4 detected keywords. The (x, r,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

The success strategy relies on the idea that "the higher the prediction certainty, the more relevant the explanation" (Bhan et al., 2023a). On the contrary, 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 an auxiliary model to target promising tokens, making it more flexible than other methods.

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 the AMPLIFY framework 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 self_exp rationales where AMPLIFY only implements attribution methods.

This makes Self-AMPLIFY more versatile. Post-hoc explanations are computed to explain each (x, y) pair and, finally, to 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 the latter 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 provides more details about this post hoc attribution computation step.

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. The self_topk explanation is generated in a "answer then generate" post hoc manner, since the text containing the k most important keywords is generated given the ground truth answer y.

Finally, self_exp consists in prompting f to generate a p-step natural language explanation in a post hoc manner, given the ground truth y. Therefore, self_exp 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 natural language explanation previously generated, and p is the number of steps in the rationale. Appendix A.3 provides more details about the prompts used to generate self_topk and self_exp rationales. Figure 4 gives an example of generated answers conditioned by different rationale generator (Self-AMPLIFY in 3 different versions, classical prompting and Auto-CoT).

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.

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 two datasets and two autoregressive SLMs. Four versions of Self-AMPLIFY, respectively based on four post hoc explanations methods, are compared to Auto_CoT, a competitor automatically generating rationales and IO, a traditional prompting setup baseline. Finally, we assess the impact of the selection strategy and the post hoc explanation method on Self-AMPLIFY.

4.1 Experimental protocol.

Datasets. Self-AMPLIFY is tested on two common LMs' benchmarks. The ARC Challenge (Clark et al., 2018) is a commonsense reasoning dataset containing grade-school science questions. ARC is designed to evaluate a model's ability to use prior knowledge about the world. The Snarks dataset (Srivastava et al., 2022) aims at distinguishing between sarcastic and non-sarcastic sentences. 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 the usual benchmarks. Mistral Tiny (Jiang et al., 2023) is a SLM with 7 billion parameters achieving state-of-the-art performance among other SLMs in a wide variety of NLP tasks. Zephyr (Tunstall et al., 2023)

Selection strategy	Method	Post hoc explainer	Mistral Tiny		Zephyr	
			ARC Challenge	Snarks	ARC Challenge	Snarks
Success	Self-AMPLIFY	DeepLift	71.7	47.6	**67.3**	42.8
		KernelShap	71.7	46.9	66.7	41.4
		Self_topk	72.7	31.7	66.7	41.3
		Self_exp	**74.3**	**64.1**	**67.3**	46.2
	IO		68.3	49.0	64.4	48.2
	Auto-CoT		73.3	53.8	63.3	**57.9**
Error	Self-AMPLIFY	DeepLift	70.7	56.6	69.3	48.3
		KernelShap	68.0	54.5	68.7	45.5
		Self_topk	70.0	57.2	**71.7**	**59.3**
		Self_exp	72.0	**68.3**	63.0	48.3
	IO		72	46.9	63.0	36.5
	Auto-CoT		**72.7**	54.5	65.0	49.7

Table 1: Self-AMPLIFY accuracy (%) on 2 test sets and comparison with competitors. Self-AMPLIFY is tested on 4 versions, depending on the post hoc explainer used to generate rationales. IO stands for "input-output" standard prompting. Auto-CoT (Zhang et al., 2022) is a competing method automatically generating rationales to enhance the input prompt.

is also a 7 billion parameters SLM demonstrating high performance on the same benchmarks.

Self-AMPLIFY versions and competitors. We test the four implemented post hoc explanation methods in Self-AMPLIFY : DeepLift, KernelShap, self_topk and self_exp. These Self-AMPLIFY versions are compared to a traditional (x, y) prompting setup (input-output, IO) and Auto-CoT (Zhang et al., 2023). For a fair comparison, we apply Self-AMPLIFY and Auto-CoT on the same n-shot sample. We do not compare Self-AMPLIFY to AMPLIFY because the main interest of our work is to generate rationale without using any additional model. Since AMPLIFY uses a proxy fine tuned LM (up to 200 epochs) to generate its rationales, the comparison would be unfair.

Post hoc attribution methods and self_topk are computed with $k = 6$, meaning that the 6 most important tokens are used to generate rationales. Self_exp p-step rationales are generated with $p = 3$, and ICL context size is set at $n = 8$.

The four versions of Self-AMPLIFY and its competitors are tested on the same samples from ARC Challenge and Snarks. Therefore, contexts are enriched from the same training corpus $\mathcal{T}$ and inference is performed on the same test sets. In particular, evaluation is performed on 350 randomly sampled texts from ARC Challenge and 145 from Snarks.

The Mistral Tiny model is used in the end to recover the SLMs response in the correct format, belonging to the label space (for example A, B, C or D for ARC Challenge) to compute accuracy.

4.2 Results.

Global Results. As shown in Table 1, in each case (dataset, SLM), the best result is obtained by one of the Self-AMPLIFY variants. Self-AMPLIFY improves performance as compared to IO and achieves better results than Auto-CoT on average, validating the value of post hoc generated rationales to enhance SLMs. Post hoc attribution methods can lead to significant performance gain as compared to IO and Auto-CoT. In particular, the Self_exp generated rationales induce generally better performance, especially with the success selection strategy. Accuracy is significantly higher on ARC Challenge than Snarks, the latter task being more challenging.

Impact of the selection strategy and the task. Table 1 shows that the selection strategy has a significant impact on the results. The selection strategy leads to wide performance gaps on the Snarks dataset, where the error selection strategy is more advantageous for Self-AMPLIFY compared to its competitors.

This gap is is even more pronounced with post

selection strategy	DeepLift	KernelShap	self_topk	random
error	67.4 ±3.8	67.4 ±3.0	**68.6** ±1.1	61.6 ±4.5
success	67.7 ±2.1	**69.0** ±3.1	67.0 ±2.0	63.9 ±6.6

Table 2: Average accuracy (%) and standard deviation computed on 6 runs of Self-AMPLIFY for different $topk$ post hoc explainers, on Mistral Tiny and ARC Challenge. All $topk$ methods do better in average than the random keyword generator, for every selection strategy.

selection strategy	topk	DeepLift	KernelShap	self_topk
error	4	70.6	67.7	**72.7**
	6	70.0	68.0	70.0
success	4	70.3	70.3	**72.7**
	6	70.3	70.3	67.7

Table 3: Average accuracy (%) of Self-AMPLIFY for different $topk$ post hoc explainers and different topk values, on Mistral Tiny and ARC Challenge. Self_topk rationales lead to better performance with 4 important keywords in the generated rationales.

hoc attribution explainers, where KernelShap and DeepLift have low performance with the success strategy whereas their accuracy becomes higher than their competitors with the error selection strategy (between 4 and 9 points difference).

We hypothesize that these disparities can be explained by the complexity of the Snark task (sarcasm detection). Assuming that "the higher the prediction certainty, the better the related explanation", post hoc attribution explanations seem to fail generating useful rationales on previously successfully predicted instance to enhance performance. The performance gain from the error selection strategy would then be related to the corrective signal added in the prompt to avoid similar misclassifications.

Self-AMPLIFY leads to better result in general on the ARC Challenge dataset with almost every type of post hoc explainer. In particular, we hypothesize that the good results obtained by post hoc attributions methods on the ARC Challenge dataset and the success strategy are due to the lower complexity of the task. Generating rationales from successfully classified texts with such a level of complexity would then provide high quality signal to improve SLM's performance.

Impact of the post hoc explainer. The Self_exp post hoc explainer generates more useful rationale to improve SLMs' accuracy in average, as shown in Table 1. DeepLift and KernelShap performances ar similar, with DeepLift having slightly better results. In particular, Self_exp is more robust than the other post hoc explainers when Self-AMPLIFY does not lead to significant performance gain compared to its competitors (see for example Mistral Tiny on Snarks with the success selection strategy).

Which $topk$ explainer leads to more useful rationales ? We perform a deeper analysis of the impact of the $topk$ post hoc explainer through an ablation study. We run 6 times Self-AMPLIFY on the same test set from ARC Challenge in order to have different ICL context samples from one run to the other. The analysis is performed on Mistral Tiny, with rationales generated from each of the 3 considered post hoc $topk$ explainers (DeepLift, KernelShap and Self_topk) and a random $topk$ keyword generator (random). This way, we assess the sensitivity of Self-AMPLIFY to the post hoc explainer generating $topk$ explanations.

Table 2 shows the results of the performed ablation study. Self-AMPLIFY leads to a better accuracy when grounded on a $topk$ explainer compared to the random baseline. Self_topk gives the highest average accuracy when samples are selected according to the success strategy. KernelShap gives the highest average accuracy when samples are selected according to the error strategy.

Impact of the number of keywords in the rationale. We run Self-AMPLIFY on the same ICL sample and test set from ARC Challenge with $k = 4$ and $k = 6$ to assess the impact of the number of keywords. The analysis is performed for each $topk$ explainer and each selection strategy. Table 3 shows the obtained results. Self_topk is the only $topk$ explainer inducing a lower accuracy with 6

keywords instead of 4, whereas KernelShap and DeepLift are very stable

5 Discussion

We introduced Self-AMPLIFY, an extension of the AMPLIFY framework, automatically generating rationales to enrich the prompt in ICL settings for small language models. 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.

The proposed framework is versatile and can work with any other post hoc attribution methods, such as Integrated Gradient or LIME. Therefore, one can chose the appropriate attribution method to use, depending on the level of information available about the model. For example, some APIs do not provide access to model internal parameters, making the use of KernelShap a better option as compared to DeepLift. In addition, it would be interesting to test Self-AMPLIFY on even smaller SLMs with less reasoning abilities such as TinyLlama (Zhang et al., 2024).

As a future work, it would be interesting to assess the faithfulness of the generated rationales. It would be also valuable to analyze in more depth the keywords generated by the SLM in ICL settings. Generated post hoc explanations could be compared to ground truth post hoc explanations that would have been computed in a post hoc manner. This would provide rich insights about the ability of LMs to learn to explain. We see these perspectives as promising paths towards a better understanding of LMs' ability to self-explain and generate faithful explanations.

A way of improvement of Self-AMPLIFY could be to generate other types of rationales to enrich the prompt such as textual counterfactual examples (see e.g. (Bhan et al., 2023b) for a recent method). By providing slight modifications to apply to the input text to change the SLM outcome, counterfactual examples could give strong correcting yet simple signals to self-enhance the SLM. It constitutes promising directions to generate valuable rationales to enhance SLM's performance.

Finally, a deeper analysis of the link between task complexity, selection strategy and Self-AMPLIFY performance would provide precious information about situations where post hoc explainers provide more valuable rationales than others.

For anonymity reasons, we do not provide a link to the Self-AMPLIFY github. The code will be available upon acceptance to facilitate reproduction and further research.

6 Conclusion

This paper presents Self-AMPLIFY, an extension of the AMPLIFY framework to build rationales from a SLM to improve itself in ICL settings. Self-AMPLIFY is the first method to generate rationales without the use of any auxiliary side model. Self-AMPLIFY implements 2 selection strategies and 4 post hoc explanation methods, making it versatile and flexible. Self-AMPLIFY results in performance gain in the two considered tasks, outperforming Auto-CoT. Finally, this work sheds light on the interest of using post hoc attribution methods to enhance SLM's performance.

7 Limitations

Datasets and models. In this work we have tested Self-AMPLIFY by applying it on 2 datasets and 2 language models. The conclusions of our work would have more weight if other datasets were included in the study. Furthermore, it would be interesting to test Self-AMPLIFY on even smaller SLMs with less reasoning abilities such as TinyLlama (Zhang et al., 2024). 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., 2022), 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.4) to make it even more affordable.

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.

References

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.

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*.

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].

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? ArXiv:2401.07927 [cs].

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.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are Emergent Abilities of Large Language Models a Mirage? ArXiv:2304.15004 [cs].

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, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

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.

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*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. ArXiv:2302.13971 [cs].

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. *arXiv preprint arXiv:2206.07682*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. ArXiv:2201.11903 [cs].

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, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. ArXiv:2305.10601 [cs].

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic Chain of Thought Prompting in Large Language Models. ArXiv:2210.03493 [cs].

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). We

will make our code available upon acceptance to facilitate reproduction and further research.

A.2 SLMs implementation Details

Small Language Models. The library used to import the pretrained SLMs is Hugging-Face. In particular, the backbone version of Mistral is `mistralai/Mistral-7B-Instruct-v0.2`, and the one of Zephyr is `HuggingFaceH4/zephyr-7b-beta`.

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>'`

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`

A.3 Prompting format

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

Prompt for `self_topk` rationale generation
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 `self_exp` 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.4 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`

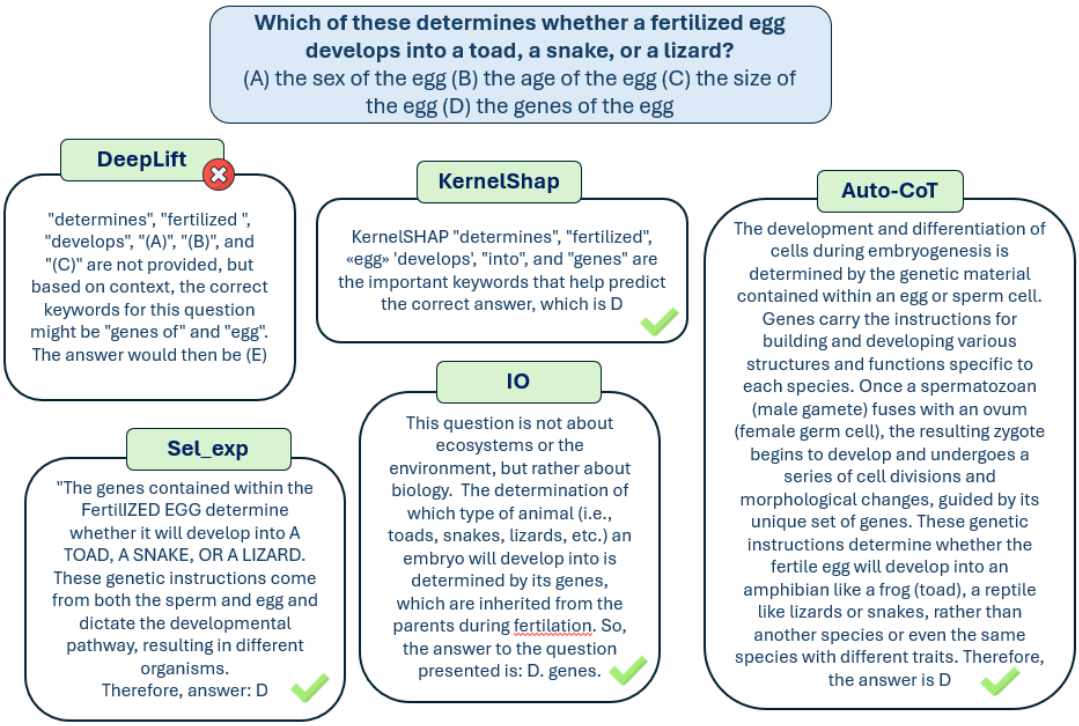

Figure 4: ARC Challenge answers conditioned by different ICL prompt built from different rationale generators.

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 an attribution equal to 0. The baseline is thus constructed as a modified version of the text on which DeepLift is applied. The only difference is that the prompt corresponding to the statement, the question part and possible answers is replaced with padding.