PINTO: FAITHFUL LANGUAGE REASONING USING PROMPTED-GENERATED RATIONALES

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural language models (LMs) have achieved impressive results on various language-based reasoning tasks by utilizing latent knowledge encoded in their own pretrained parameters. To make this reasoning process more explicit, recent works retrieve a rationalizing LM's internal knowledge by training/prompting it to generate free-text rationales, which can be used to guide task predictions made by either the same LM or a separate reasoning LM. However, rationalizing LMs require expensive rationale annotation, without any assurance that the generated rationales improve LM task performance or faithfully reflect LM decision-making. In this paper, we propose PINTO, an LM pipeline that *rationalizes* via prompt-based learning, and learns to *faithfully reason over rationales* via counterfactual regularization. First, PINTO maps out a suitable reasoning process for the task input by prompting a frozen rationalizing LM to generate a free-text rationale. Second, PINTO's reasoning LM is fine-tuned to solve the task using the generated rationale as context, while regularized to output less confident predictions when the rationale is perturbed. Across four datasets, we show that PINTO significantly improves the generalization ability of the reasoning LM, yielding higher performance on both in-distribution and out-of-distribution test sets. Also, PINTO leverages the rationales more faithfully than competitive baselines do.[1]

1 INTRODUCTION

Many language-based reasoning tasks require retrieving and reasoning over knowledge beyond the task input—*e.g.,* commonsense reasoning and closed-book QA (Fig. 1, left) (Talmor et al., 2018; Mihaylov et al., 2018). Neural language models (LMs) have achieved impressive results on such tasks by utilizing latent knowledge encoded in their pretrained parameters (Raffel et al., 2020b; Brown et al., 2020). Still, given LMs' black-box nature, it is unclear whether this knowledge is being used properly (Doshi-Velez & Kim, 2017; Lipton, 2018). Previous studies have shown that LMs often learn spurious correlations from artifacts in downstream training data, thus limiting their generalizability (Branco et al., 2021; Geirhos et al., 2020; D'Amour et al., 2020).

With this in mind, a number of prior works aim to make LMs' reasoning processes more *explicit* by generating free-text rationales, which use LMs' internal knowledge to describe a reasoning process in natural language (Narang et al., 2020; Wei et al., 2022b; Marasović et al., 2022; Zelikman et al., 2022). In the *fine-tuned self-rationalizing* paradigm, a single LM is fine-tuned to jointly generate the task output and rationale (Narang et al., 2020; Marasović et al., 2022; Zelikman et al., 2022). In the *prompted self-rationalizing* paradigm, a single LM is instead frozen and prompted to jointly generate the task output and rationale, with the prompt consisting of a few input-output-rationale demonstrations (Wei et al., 2022b). In the *pipeline-rationalizing* paradigm, a fine-tuned LM first generates the rationale, which is then used as input for a separate fine-tuned reasoning LM to generate the output (Kumar & Talukdar, 2020; Rajani et al., 2019).

However, when considering generalization performance, reliability, and deployment cost, these existing paradigms all have key limitations. Fine-tuned self-rationalizing LMs often perform worse than non-rationalizing LMs, since their parameters are learned using two relatively dissimilar objectives, while also requiring expensive rationale annotations (Wiegreffe et al., 2020; Narang et al.,

[1] Code and data have been submitted and will be publicly released after the review period.

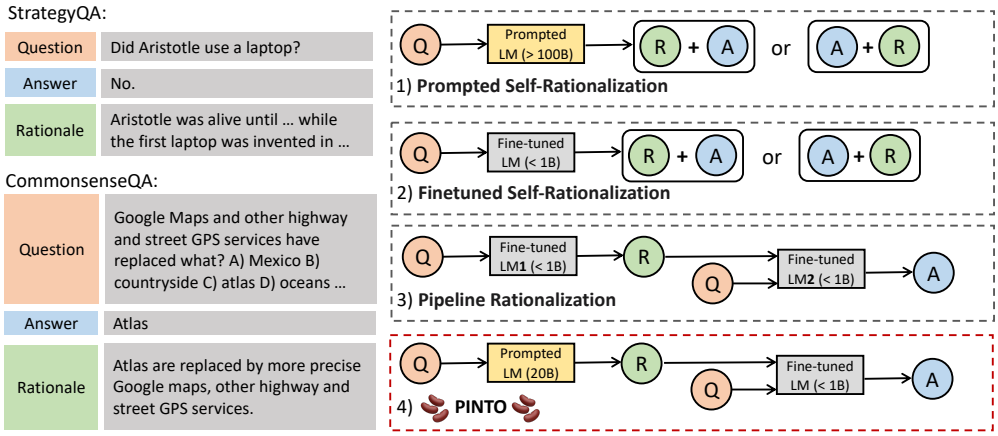

Figure 1: **Rationale-Based Language Reasoning.** (a) Examples of reasoning tasks that require implicit knowledge beyond task inputs. (b) Comparison of existing paradigms for providing free-text rationales along with predictions.

2020). Prompted self-rationalizing LMs yield strong task performance and only need a few rationale demonstrations for the prompt, but are computationally prohibitive since they generally require very large-scale (*i.e.,* over 100B parameters) LMs to work effectively (Wei et al., 2022a;b). Besides requiring expensive rationale annotations, pipeline-rationalizing LMs' generated rationale forms a non-differentiable bottleneck between the two modules, which complicates end-to-end training and can hurt task performance (Wiegreffe et al., 2020; Hase et al., 2020). Moreover, none of these paradigms has a mechanism for regularizing the rationale generation to *faithfully* reflect the reasoning process of the LM, without hurting task performance.

In this paper, we propose **P**rompted Rat**I**onalizing with Cou**NT**erfactual Reas**O**ning (**PINTO**), an LM pipeline that rationalizes via prompt-based learning, then reasons over the task input and rationale via counterfactual regularization. PINTO's *rationalizing module* is a medium-scale (*i.e.,* 20B parameters) LM that contains vast latent knowledge obtained via pretraining (Black et al., 2022). Though prohibitive to fine-tune, it is affordable for prompt-based learning. Given the task input and a minimal input-output demonstration prompt, the rationalizing module uses its internal knowledge to map out a suitable reasoning process for the task input by generating a free-text rationale. The rationalizing module is frozen during fine-tuning, which drastically reduces training costs and prevents it from exploiting spurious shortcuts in the downstream training data. PINTO's *reasoning module* is a small-scale (*i.e.,* under 1B parameters) LM to which knowledge is transferred from the much larger rationalizing module. The reasoning module is fine-tuned to solve the downstream reasoning task by using the generated rationale as context for the task input. Crucially, to help ensure that the rationale dictates the behavior of the reasoning module, the reasoning module is regularized to output less confident predictions when the rationale is perturbed. To simulate shortcut reasoning with rationales, we consider two rationale perturbation strategies: token masking (*i.e.,* rationale is ignored) and token replacement (*i.e.,* rationale is misused).

Across four question answering datasets (CSQA, StrategyQA, OpenBookQA, QASC), we show that PINTO significantly improves the reasoning LM's generalization, yielding higher performance on both in-distribution (ID) and out-of-distribution (OOD) test sets. Also, we find that rationales are leveraged more faithfully by PINTO than by other methods, leading to better performance in low-resource settings. Furthermore, we show that PINTO's counterfactual regularization improves the reasoning module's robustness to noise in the rationalizing module's generated rationales.

2 Rationale-Based Language Reasoning

In this work, we study LMs' ability on language-based reasoning using implicit knowledge. We consider a specific type of multi-choice question answering (QA) tasks where the required knowledge for answering the question is not explicitly provided in the input and needs to be inferred from the LM's parameters (Talmor et al., 2019; Khot et al., 2020): Given a question q and a set of answer

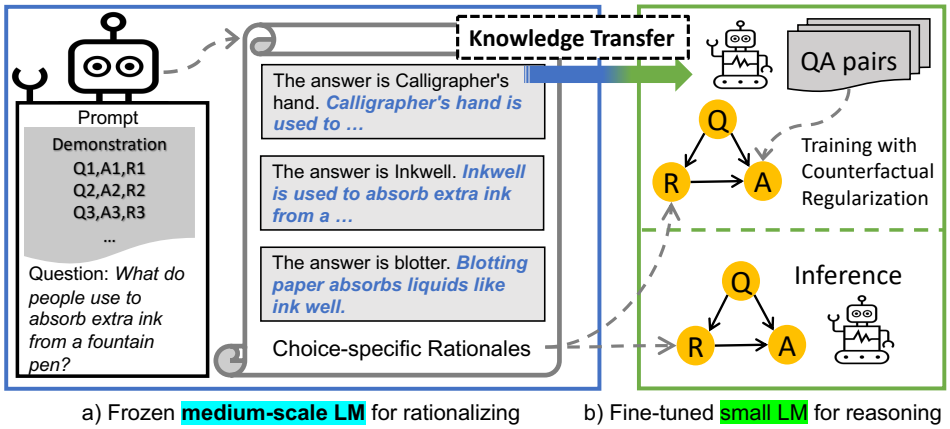

Figure 2: **Overview of PINTO.** (1) A frozen medium-scale LM is prompted to generate choice-specific rationales. (2) A small-scale LM is fine-tuned to reason over the generated rationales. (3) We introduce counterfactual regularization in addition to standard training loss to ensure the rationales are leveraged properly. During inference, the rationalizing LM is prompted with a new question to generate rationales, which are provided to the reasoning module to make a prediction.

choices $A = \{a_i\}$, the model's goal is to predict a plausibility score $\rho(q, a_i)$ for each (q, a_i) pair, so that the predicted answer $\hat{a} = \arg\max_{a_i \in A} \rho(q, a_i)$ matches the correct answer choice $a^* \in A$.

Motivated by the LM tendency to exploit reasoning shortcuts for solving tasks (Branco et al., 2021), we focus on methods that explicitly generate free-text rationales to explain their predictions. Whereas extractive rationales are limited to input token scoring (Denil et al., 2014; Sundararajan et al., 2017; Chan et al., 2022), free-text rationales use natural language to describe a reasoning process (*e.g.,* things beyond the task input) (Narang et al., 2020; Wei et al., 2022b). Below, we discuss several paradigms (see also Fig. 1) for rationale-based language reasoning.

Fine-Tuned Self-Rationalization In this paradigm, an LM is *fine-tuned* to autogregressively generate the task output and rationale as a single sequence (Narang et al., 2020; Liu et al., 2018). If the rationale is generated after the task output, then the rationale is conditioned on the task output, and vice versa. Since the LM parameters are shared across two relatively dissimilar objectives, they often perform worse than non-rationalizing LMs (Wiegreffe et al., 2020; Narang et al., 2020). Notably, this paradigm requires expensive rationale annotations for all training instances.

Prompted Self-Rationalization In this paradigm, a pretrained LM is *frozen* and *prompted* to autogregressively generate the task output and rationale as a single sequence, with the prompt consisting of a few input-output-rationale demonstrations (Lampinen et al., 2022; Wei et al., 2022b). If the rationale is generated after the task output, then the rationale is conditioned on the task output, and vice versa. This paradigm performs well and it only needs a few rationale annotations for the prompt, but it is computationally prohibitive since it generally requires very large-scale (*i.e.,* over 100B parameters) LMs to work effectively (Lampinen et al., 2022; Wei et al., 2022b).

Pipeline Rationalization Here, a fine-tuned rationalizing LM first generates the rationale, which is then used as input for a separate fine-tuned reasoning LM to predict the task output (Kumar & Talukdar, 2020; Rajani et al., 2019). This paradigm's generated rationale forms a discrete (*i.e.,* non-differentiable) bottleneck between the two modules, which complicates end-to-end training and can hurt task performance (Wiegreffe et al., 2020; Hase et al., 2020). Additionally, the dedicated rationalizing LM requires extra rationale annotation/computation costs.

3 PINTO: Faithful Language Reasoning Using Prompted-Generated Rationales

PINTO is a two-stage, rationalize-then-reason pipeline, designed to address the limitations of existing paradigms for rationale-based language reasoning (§2). Like the pipeline rationalization paradigm, PINTO has separate modules for rationalizing and reasoning. However, PINTO's rationalizing module is prompted instead of fine-tuned. As a result, PINTOdoes not suffer from the non-differentiable bottleneck issue and has lower rationale annotation/computation costs.

Table 1: **Rationalization Prompts.** The format of our prompts for rationalization with a medium-scale LM. The prompt consists of a few examples as demonstration on how to rationalize for a question-choice pair and placeholders for new question and a target choice.

Task	CommonsenseQA	OpenBookQA
Prompt	**Q**: What do people use to absorb extra ink from a fountain pen? **Answer Choices**: (a) shirt pocket (b) calligrapher's hand (c) inkwell (d) desk drawer (e) blotter **A**: The answer is blotter. *Blotting paper absorbs liquids like ink well.*	**Q**: How do you reduce pollution? **Answer choices**:(a) igniting fuel and oxidiser (b) transportation technology ... (h) using less resources **A**: The answer is using less resources. *Conserving resources has a positive impact on the environment. Use of resources affects the environment such as pollution.*

Following prior works, PINTO is based on choice-specific rationales (Kumar & Talukdar, 2020; Hase et al., 2020). First, given q and A, the *rationalizing* module generates a set of choice-specific rationales $R = \{r_i\}$, where each r_i explains a reasoning process that supports answer choice $a_i \in A$ (§3.1), as opposed to generating one rationale per question. We opt for this design choice because rationales are often answer-leaking (Sun et al., 2022), *i.e.,* the rationale itself is already sufficiently predictive of one of the answer choices. If the rationalizing module only generates one rationale per question, then it is forced to make an "early decision", and the reasoning module would only be left to recover the answer from the rationale (Kumar & Talukdar, 2020). While prior works require expensive rationale annotations to train/prompt the rationalizing module (Kumar & Talukdar, 2020; Hase et al., 2020), PINTO's rationalizing module is a frozen pretrained LM that uses only a few question-answer demonstrations as a prompt (§3.1). Second, given q, $a_i \in A$, and $r_i \in R$, the *reasoning* module outputs plausibility score $\rho(q, a_i, r_i)$ (§3.2). We also design a regularization objective that encourages the reasoning module to properly use the rationales to predict the answer (§3.3). We describe each module in more detail below.

3.1 Rationalizing Module

Prior works mainly rely on human-annotated rationales for teaching a model to rationalize (Kumar & Talukdar, 2020; Hase et al., 2020; Sun et al., 2022). However, such rationale annotations are expensive and frequently of low quality (Aggarwal et al., 2021; Sun et al., 2022; Rajani et al., 2019), *e.g.,* not providing sufficient commonsense knowledge to support a given answer. Meanwhile, a recent study shows that rationales automatically generated by pretrained LMs are often preferred over human-annotated rationales (Wiegreffe et al., 2021). Therefore, for PINTO's rationalizing module, we propose using a pretrained LM to generate rationales via in-context learning, which prompts the frozen LM to retrieve knowledge from its parameters (Wei et al., 2022b).

The prompt consists of a fixed set of question-answer demonstrations that are randomly selected from the training set. Each demonstration consists of a question q, answer choices A,[2] gold answer $a^* \in A$, and a human-annotated free-text rationale $r^* \in R$ for a^* (Table 1).[3] With this prompt p, we use the LM to generate rationales for every instance from the dataset. Specifically, for each $a_i \in A$ of some instance (q, A), the rationalizing LM's input is constructed as [p, q, A, a_i]. Then, we use greedy decoding of the LM output to obtain rationale r_i for a_i. Note that the LM input does not have any information about the gold answer a^*. Our design of the rationalizing module assumes that r_i will be aligned with common sense if and only if $a_i = a^*$, since it should intuitively be difficult to retrieve correct commonsense knowledge that supports an incorrect answer choice. The reasoning module then predicts the correct answer by reasoning over the rationales for each answer choice.

3.2 Reasoning Module

Given a question q, the answer choices A, answer candidate $a_i \in A$, and rationale r_i, the reasoning module learns to output plausibility score $\rho_i = \rho(q, A, a_i, r_i)$. Following prior works, we use a text-to-text Transformer LM as the backbone of our reasoning module (Wiegreffe et al., 2020; Hase et al., 2020). For each a_i, the reasoning module's input is defined as the token sequence $s = [q \oplus a_1 \oplus ... \oplus a_{|A|} \oplus r_i]$, where $\oplus$ denotes concatenation. Meanwhile, the reasoning module's

[2]We include the answer choices A in the prompt so that the LM is aware of all the available choices and thus could generate a rationale that is more distinctive.

[3]As opposed to full human annotation, we only need a few (usually <8) examples for one dataset.

output is obtained by sequentially teacher-forcing a_i's tokens $t_i = [t_1^i, t_2^i, ..., t_{|a_i|}^i]$ into the decoder, rather than via greedy decoding. This way, we can compute the reasoning module's output token probabilities for arbitrary answer choices a_i. Following Shwartz et al. (2020), we compute a_i's plausibility score ρ_i by aggregating t_i's token probabilities as:

$$\rho_i = \frac{1}{|a_i|} \sum_{j=1}^{|a_i|} P(t_j^i \,|\, t_{j-1}^i, ..., t_2^i, t_1^i, q, A, r_i).$$

Next, we use the softmax function to normalize ρ_i as probability $P(a_i \,|\, q, A, R) = e^{\rho_i} / \sum_{j=1}^{|A|} e^{\rho_j}$. During **inference**, the rationalizing module firstly generate $R = \{r_i\}$ given a new question and answer choices, and then the predicted answer choice is computed by the reasoning module as $\hat{a} = \arg\max_{a_i \in A} P(a_i \,|\, q, A, R)$.

3.3 Training

For a multi-choice QA, the standard training objective is to maximize the likelihood of the correct answer choice using cross-entropy loss, computed as:

$$\mathcal{L}_{\text{std}} = -\sum_{a_i \in A} Q(a_i \,|\, q, A) \log P(a_i \,|\, q, A, R), \tag{1}$$

where $Q(a_i \,|\, q, A)$ is 1 if $a_i = a^*$ and 0 otherwise. Let $Q(A \,|\, q, A)$ be the one-hot target distribution over all $a_i \in A$. There can be spurious correlations between q and A (Branco et al., 2021), so the reasoning module may take undesirable shortcuts instead of properly using the rationale to predict the answer (Gururangan et al., 2018; McCoy et al., 2019). In this case, the rationales would be unfaithful in explaining the model's behavior and useless for model debugging.

To address this, we introduce a counterfactual regularization objective in which the reasoning module is regularized to output less confident predictions when the rationale is not leveraged properly (*i.e.,* shortcuts are used). This is implemented using label smoothing (Szegedy et al., 2016), which softens the target distribution $Q(A \,|\, q, A)$ by linearly combining it with a noisy distribution $U(A \,|\, q, A)$, often set as the uniform distribution. Therefore, given tunable label smoothing factor $0 < \epsilon < 1$, we compute the label-smoothed target distribution as: $Q'(A \mid q, A) = (1 - \epsilon)\, Q(A \,|\, q, A) + \epsilon\, U(A \,|\, q, A)$.

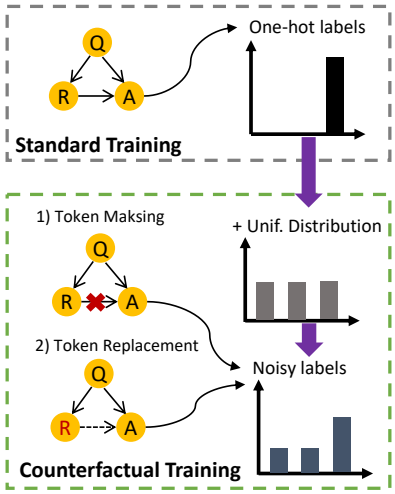

Figure 3: **Standard Training vs. Counterfactual Training.** For counterfactual regularization, we train the reasoning module with noisy labels when the rationale tokens are either masked or replaced.

In order to simulate shortcut reasoning, we consider the two strategies for perturbing the generated rationales r_i. **Token Masking** addresses the case where the reasoning module ignores the rationale and instead exploits spurious cues in the rest of the input. To simulate this, we mask the rationales from the input. Recall that the backbone of the reasoning module is a Transformer LM, which uses a self-attention mechanism to aggregate information across tokens. Hence, we implement rationale masking by zeroing the attention mask for rationale tokens.**Token Replacement** addresses the scenario in which the reasoning module misunderstands the rationales. It randomly replaces $k\%$ of the rationale tokens with other tokens randomly sampled from the entire vocabulary.

At each fine-tuning step, we randomly select one of the strategies for obtaining perturbed rationales $R' = \{r_i'\}$, which helps keep the LM from overfitting to any particular strategy. Then, the counterfactual regularization loss is computed as:

$$\mathcal{L}_{\text{c-reg}} = -\sum_{a_i \in A} Q'(a_i \,|\, q, A) \log P(a_i \,|\, q, A, R'). \tag{2}$$

This counterfactual regularization teaches the reasoning module to be less confident when the rationales are either absent or problematic, so that it can learn to make sounder use of the rationales.

4 EXPERIMENTAL SETUP

Questions and hypotheses We design experiments to answer the following questions: (1) What is the impact of our PINTO pipeline on faithfulness and end-task performance? We expect our pipeline with counterfactual training technique to obtain improvements in both aspects. (2) How does the quality of rationales affect the end-task performance of PINTO? We hypothesize that improving the quality of the rationales of PINTO improves its accuracy. (3) Does faithful reasoning based on rationales lead to better generalization? We expect that a method like PINTO that learns to rely on rationales can better generalize to a low resource setting and out-of-distribution (OOD) datasets.

Datasets We experiment with several CSR benchmarks. (1) CommonsenseQA (Talmor et al., 2018) is a 5-choice QA dataset testing general commonsense reasoning about the concepts from ConceptNet (Speer et al., 2017). (2) StrategyQA (Geva et al., 2021) is a binary (yes/no) QA dataset that requires models to infer the reasoning strategy. (3) OpenBookQA (Mihaylov et al., 2018) is a 4-choice QA dataset that requests reasoning based on open book as well as broad commonsense knowledge. (4) QASC (Khot et al., 2020) is an 8-choice QA dataset that requires a system to answer a question with a valid composition of basic facts using common sense. Since the gold labels for the testing sets of these datasets are not publicly available, we treat the official development set as our test set, and separate the training data into our own training set and development set.

Implementation Details For the rationalizing module, we use GPT-neox (Black et al., 2022), a pretrained, autoregressive LM with 20B parameters. We manually annotate 7 examples to set up the prompt for each task dataset. For the reasoning module, we adopt T5-base (Raffel et al., 2020a) with only 220 million parameters, which is around two orders of magnitude smaller than the rationalizing module. During fine-tuning, the standard training loss (Eq. 1) and our counterfactual training loss (Eq. 2) are directly combined as the overall training loss. For perturbing rationales, we randomly choose the token masking or token replacement strategy with a equal chance in each training batch. The replacing rate for token replacement is empirically set to 30%. We run all the experiments on the compared methods 4 times using a fixed set of random seeds and report the average results.

Baselines (1) *Without Rationales* is a T5-based model fine-tuned with the task dataset without using any rationales as additional input. (2) *Prompted Self-Rationalization* is a frozen LM which learns from a few examples in the prompt to firstly generate a few short sentences as the rationale and then predict the answer. We experiment with GPT-neox (20B) and also report the results of GPT-3 (175B) from Wei et al. (2022b). (3) Distilled Self-Rationalization is a small LM (T5-base) trained on the rationales generated by the Prompted Self-Rationalization model. We implement two variants of the distillation model: a) Rationalize-First, which firstly generates the rationale and then predicts the answer, and b) Predict-First, which firstly predicts the answer and then generates the rationale. (4) *Standard Training* adopts the same rationalizing and reasoning pipeline as our method, but the reasoning module is not fine-tuned with the counterfactual training loss. (5) *Dropout Context* is the same as the Standard Training baseline except that during the fine-tuning of the reasoning module, the question is randomly dropped out from the input, which is a strategy adopted in prior work (Hase et al., 2020) to encourage the reasoning module to make good use of the input rationales.

Further, we also consider two variants of PINTO, namely *Token Masking Only* and *Token Replacement Only* as baselines. These baselines only adopt token masking or token replacement for perturbing rationale tokens, respectively.

Evaluation Metrics To evaluate *task performance*, we use accuracy as the main metric and consider both ID and OOD evaluation sets in our experiments. To evaluate the *faithfulness* of the generated rationale to the model's prediction, we adopt LAS metric (Hase et al., 2020), which measures how well rationales help a simulator predict a model's output. We implement the simulator with a fine-tuned T5-base model as recommended by Hase et al. (2020).

5 EXPERIMENTS

5.1 MAIN RESULTS

Performance on ID data Table 2 shows the task performance in accuracy of all the compared methods on the four CSR datasets we consider, from which we make two observations. First, the Prompted Self-Rationalization baseline with GPT-neox (20B) generally does not outperform the

Table 2: **ID Results.** Task performance (accuracy) and faithfulness (LAS) of the compared methods on the testing datasets. For Pipeline models, the rationalizing module is GPT-neox (20B) while the reasoning module for the fine-tuning methods is T5-based. We bold the results that exceed the second best with statistical significance (p-value< 0.05).

	CSQA		StrategyQA		OpenBookQA		QASC	
Method	Acc.↑	LAS↑	Acc.↑	LAS↑	Acc.↑	LAS↑	Acc.↑	LAS↑
w/o Rationales	58.68	-	58.12	-	55.85	-	35.58	-
Prompted Self-Ra. (GPT-neox)	38.41	11.66	55.31	1.09	33.80	14.67	32.61	32.01
Prompted Self-Ra. (GPT-3)	**73.50**	1.38	**66.53**	0.60	-	-	-	-
Distill. Self-Ra. (Explain-First)	51.97	11.30	50.20	1.29	48.90	13.76	33.34	31.82
Distill. Self-Ra. (Predict-First)	55.77	6.86	54.61	-2.68	50.25	12.30	34.53	18.48
Standard Training	59.48	18.75	57.11	1.50	56.65	17.03	37.50	37.91
Dropout Context	59.64	20.40	51.45	0.62	57.55	**18.76**	35.37	37.54
PINTO	61.67	**24.22**	60.87	**3.35**	**58.85**	18.02	37.82	**38.98**
- Token Masking Only	60.46	17.44	59.12	1.74	58.35	13.06	37.39	34.06
- Token Replacement Only	60.38	22.54	58.72	2.11	58.10	18.01	37.47	34.61

Table 3: **OOD Results.** Performance (accuracy) of the compared methods, which are firstly trained on a source dataset and then directly predict on a target dataset (denoted as $source \rightarrow target$).

Method	CS→OB	CS→QASC	OB→CS	QASC→CS	QASC→OB
w/o Rationales	32.05	39.17	24.87	45.74	34.90
Distill. Self-Ra. (Explain-First)	24.85	31.43	23.05	43.16	31.55
Distill. Self-Ra. (Predict-First)	25.10	32.26	26.43	45.17	30.50
Standard Training	31.05	40.04	25.37	47.71	34.50
Dropout Context	32.30	38.85	23.01	44.27	32.90
PINTO	**34.90**	**42.25**	**27.66**	**48.03**	**35.75**

fine-tuning methods while the GPT-3 version is reported to achieve 73.50 and 66.53 in accuracy on CSQA and StrategyQA respectively (Wei et al., 2022b). This validates that Prompted Self-Rationalization requires very large LMs to work effectively (Wei et al., 2022a). Second, simply augmenting the reasoning module with rationales (as in Standard Training) does not always lead to better results compared with the Without Rationales baseline since the rationales may not be properly utilized. The Dropout Context baseline helps to address this issue in some, but not all cases, while PINTO consistently yields the best accuracy in most of the cases.

Generalizability to OOD data To demonstrate the generalizability brought by faithful reasoning over rationales, we further investigate the performance of our method on out-of-distribution (OOD) data. The intuition is that by utilizing rationales faithfully rather than fitting only the in-distribution training data, our model achieves better OOD generalization without any fine-tuning. Table 3 shows the OOD performance of all the fine-tuning methods. We conclude that rationales are helpful in improving the generalizability of the model to a dataset unseen during fine-tuning. Among all the methods utilizing rationales, our method yields the best OOD performance, which confirms the benefit of faithful reasoning.

Rationale-Label Association Table 2 also reports the faithfulness of all the methods involving rationalization measured by LAS. We observe that PINTO achieves a much higher score compared with the baselines except on OpenBookQA. This demonstrates that with counterfactual regularization, the reasoning module can make predictions more faithfully with regard to the rationales.

5.2 Performance Analysis

How do different perturbation strategies contribute to the overall performance? Table 2 shows the results of the ablation study where we only conduct Token Masking or Token Replacement when perturbing the rationale tokens. From more cases, we note that Token Replacement leads to both better accuracy and faithfulness compared with Token Masking. This is because Token Replacement perturbs the semantics of the rationales more severely, thus further forcing the reasoning module

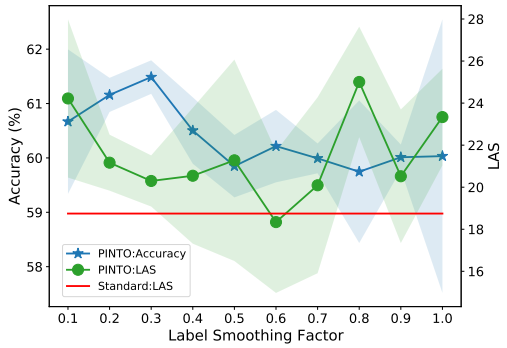

Figure 4: **Smoothing Factor Analysis.** PINTO's accuracy (left axis) and faithfulness (right axis) on CSQA, as a function of label smoothing factor.

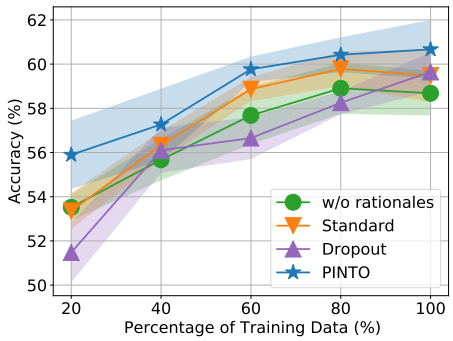

Figure 5: **Low-Resource Learning.** Performance (accuracy) of different fine-tuned models in low-resource settings on CSQA.

to properly make use of the rationales. Our method yields the best results when both types of perturbation are conducted, which validates that these two strategies consider comprehensively the different ways in which a reasoning module could use the rationales improperly.

How does the smoothing factor ϵ affect accuracy and faithfulness? Intuitively, the smoothing factor ϵ in §3.3 controls how much we want to enforce the reasoning module to consider the rationales. To understand its role in affecting task accuracy and rationales faithfulness, we conduct an analysis where we vary the ϵ from 0.1 to 1.0 when fine-tuning the reasoning module. Figure 4 shows accuracy on the left axis and faithfulness on the right axis. We can see that with a small ϵ (0.1), our method strikes a balance between obtaining high accuracy and ensuring faithfulness. When $\epsilon = 0.8$, our method yields the highest faithfulness measure at the cost of accuracy. Note that there is no clear trend in the results plot due to the randomness brought by the training dynamics.

Can faithful rationales lead to better low-resource performance? We also investigate whether, with counterfactual training, the reasoning module can be fine-tuned with less training data. Figure 5 shows the accuracy of all the fine-tuning methods. We can observe that our method consistently outperforms the baselines at different percentages of training data. The observed larger performance gap is larger when less training data is used, demonstrating the data efficiency of our method.

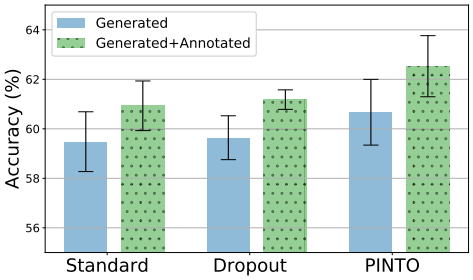

Figure 6: **Rationale Quality Analysis.** Accuracy of models with both generated and annotated rationales vs. models using only generated rationales on CSQA.

Can we refine the reasoning behavior via rationales? One important application of faithful reasoning is that rationales provide a way to refine the behavior of a model, i.e., we can correct reasoning mistakes by providing a better rationale. To verify this, we make use of ECQA (Aggarwal et al., 2021) which augments CSQA with human-annotated rationales. We directly provide the human-annotated rationales to the fine-tuned reasoning modules to obtain its oracle results, shown in Figure 6. We see that human-annotated rationales generally lead to performance gain for all fine-tuning methods whereof the gain of our method is the largest. This again showcases the merits of ensuring the faithful reasoning on rationales in refining a system.

Is our method more sensitive to perturbed rationales? Intuitively, higher rationale faithfulness (i.e., stronger connection between the rationale the and reasoning module's behavior) should yield greater sensitivity to noisily perturbed rationales. In other words, higher performance drop (sensitivity) signals higher faithfulness. To verify this, we conduct a stress test. We choose CSQA and OpenBookQA and replace each question in the testing set with a randomly sampled question but still keep the original answer choices. We then prompt our rationalizing module with the replaced question and the original choices to obtain a set of perturbed rationales. We finally provide the perturbed rationales to the reasoning module. Our results in Table 4 show that PINTO achieves a

Table 4: **Sensitivity to Noisy Rationales.** We use perturbed rationales during inference as a stress test and report the performance drop of the compared methods. .

Model	CSQA	OBQA
Standard Training	0.88	0.35
Dropout Context	2.06	0.55
PINTO	**2.62**	**1.55**

Table 5: **Case Study.** Generated rationales and model predictions from CSQA and StrategyQA.

Questions and Choice-specific Rationales	Standard	PINTO
Q: If you have leftover cake, where would you put it? (a) quandry: *Quandry is a place where you put things that you don't know ...* (b) refrigerator: *Refrigerator is used to store food.* (c) oven: *Oven is used to bake cakes.*	oven ✗	refrigerator ✓
Q: Has every astronaut survived their space journey? yes: *Astronauts have died in space. For example, the Challenger astronauts died in the Challenger space shuttle explosion.* no: *The first astronaut to die was Yuri Gagarin. Gagarin died in a plane crash.*	yes ✗	no ✓

significantly higher performance drop than the other two methods (esp. on OBQA), indicating that counterfactual regularization is effective in improving rationale faithfulness.

5.3 Case Study

We provide concrete examples in Table 5 to showcase how our prompted LM rationalizes for correct and incorrect choices and how PINTO reasons more faithfully compared with the Standard baseline. In the question (second row) from CSQA, we can see that for incorrect choices, the generated rationales do not support them to be the answer while the one for the correct choice *refrigerator* does. In the question (third row) from StrategyQA, the rationale for the correct choice *yes* is sound and reasonable while the rationale for the incorrect choice *no* is factually correct but does not answer the question directly (*died in a plane crash* vs. *died in the space journey*). For both questions, PINTO properly leverages the rationales and make the correct predictions while the Standard baseline fails.

6 Related Work

Extensive work have been done on solving implicit reasoning tasks by augmenting the models with external knowledge. They retrieve such knowledge from (1) knowledge graphs (Lin et al., 2019; Feng et al., 2020), (2) web corpus (Lv et al., 2020) or (3) pre-trained language models (Shwartz et al., 2020; Liu et al., 2021). Although the retrieved knowledge is shown to be helpful in boosting task performance, it may not necessarily explain the decisions made by their models. We focus on rationales in this work as a way to both improve a model's performance and support its decisions.

Traditional works in explaining NLP models study extractive rationales such as attention scores (Bahdanau et al., 2014) or input gradients (Sundararajan et al., 2017), which highlight the tokens that are more important for models' prediction. Extractive rationales fall short in implicit reasoning tasks since the required knowledge is not even part of the input. In contrast, free-text rationales provide a more flexible and human-accessible way of revealing models' behavior.

Prior work in generating free-text rationales can be grouped into two paradigms. The first one is self-rationalizing, which adopts one model to generate rationales and predictions together. The second paradigm is pipeline-rationalizing, where the rationalizing and reasoning modules are two separate models. Existing works (Kumar & Talukdar, 2020; Hase et al., 2020; Sun et al., 2022) use human-annotated rationales to fine-tune the rationalizing module while our PINTO uses a prompted LM to obtain those rationales without expensive human labor.

7 Conclusion

This paper presents PINTO, an LM pipeline that rationalizes with prompt-based learning and reasons via counterfactual regularization. Through prompting, we remove the need for expensive human annotation and leverage the massive knowledge encoded in a medium-sized LM to perform rationalization. With counterfactual regularization in addition to standard training objective, our reasoning module learns to reason over the generated rationales more faithfully. Experiments show that our method outperforms baselines on both in-distribution and out-of-distribution datasets in accuracy, while providing higher faithfulness. Our analysis also shows that we can further improve task performance with a more faithful reasoning module and refined rationales.

REFERENCES

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. Explanations for commonsenseqa: New dataset and models. In *Workshop on Commonsense Reasoning and Knowledge Bases*, 2021.

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

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of the ACL Workshop on Challenges & Perspectives in Creating Large Language Models*, 2022. URL https://arxiv.org/abs/2204.06745.

Ruben Branco, António Branco, Joao Rodrigues, and Joao Silva. Shortcutted commonsense: Data spuriousness in deep learning of commonsense reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1504–1521, 2021.

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

Aaron Chan, Maziar Sanjabi, Lambert Mathias, Liang Tan, Shaoliang Nie, Xiaochang Peng, Xiang Ren, and Hamed Firooz. Unirex: A unified learning framework for language model rationale extraction. *arXiv preprint arXiv:2112.08802*, 2022.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

Misha Denil, Alban Demiraj, and Nando De Freitas. Extraction of salient sentences from labelled documents. *arXiv preprint arXiv:1412.6815*, 2014.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. Scalable multi-hop relational reasoning for knowledge-aware question answering. *arXiv preprint arXiv:2005.00646*, 2020.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*, 2018.

Peter Hase, Shiyue Zhang, Harry Xie, and Mohit Bansal. Leakage-adjusted simulatability: Can models generate non-trivial explanations of their behavior in natural language? *arXiv preprint arXiv:2010.04119*, 2020.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. QASC: A dataset for question answering via sentence composition. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8082–8090. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6319.

Sawan Kumar and Partha Talukdar. Nile: Natural language inference with faithful natural language explanations. *arXiv preprint arXiv:2005.12116*, 2020.

Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X Wang, and Felix Hill. Can language models learn from explanations in context? *arXiv preprint arXiv:2204.02329*, 2022.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *arXiv preprint arXiv:1909.02151*, 2019.

Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

Hui Liu, Qingyu Yin, and William Yang Wang. Towards explainable nlp: A generative explanation framework for text classification. *arXiv preprint arXiv:1811.00196*, 2018.

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. Generated knowledge prompting for commonsense reasoning. *arXiv preprint arXiv:2110.08387*, 2021.

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8449–8456, 2020.

Ana Marasović, Iz Beltagy, Doug Downey, and Matthew E Peters. Few-shot self-rationalization with natural language prompts. *NAACL*, 2022.

R Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *arXiv preprint arXiv:1902.01007*, 2019.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. Wt5?! training text-to-text models to explain their predictions. *arXiv preprint arXiv:2004.14546*, 2020.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020a. URL `http://jmlr.org/papers/v21/20-074.html`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020b.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361*, 2019.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Unsupervised commonsense question answering with self-talk. *arXiv preprint arXiv:2004.05483*, 2020.

Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

Jiao Sun, Swabha Swayamdipta, Jonathan May, and Xuezhe Ma. Investigating the benefits of free-form rationales. *arXiv preprint arXiv:2206.11083*, 2022.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 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)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://www.aclweb.org/anthology/N19-1421.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022b.

Sarah Wiegreffe, Ana Marasović, and Noah A Smith. Measuring association between labels and free-text rationales. *arXiv preprint arXiv:2010.12762*, 2020.

Sarah Wiegreffe, Jack Hessel, Swabha Swayamdipta, Mark Riedl, and Yejin Choi. Reframing human-ai collaboration for generating free-text explanations. *arXiv preprint arXiv:2112.08674*, 2021.

Eric Zelikman, Yuhuai Wu, and Noah D Goodman. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022.

A Appendix

A.1 Experiments with RoBERTa as the reasoning module

Table 6-7 show both the ID and OOD results based on RoBERTa-Large as the reasoning module. The observations are consistent with Table 2-3 where we fine-tune T5-Base as the reasoning module.

Table 6: **ID Results.** Task performance (accuracy) and faithfulness (LAS) of the compared methods on the testing datasets. The rationalizing module is GPT-neox (20B) while the reasoning module for the fine-tuning methods is RoBERTa-Large. We bold the results that outperform the second-best method with statistical significance ($p < 0.05$).

	CSQA		**StrategyQA**		**OBQA**		**QASC**	
Method	Acc.↑	LAS↑	Acc.↑	LAS↑	Acc.↑	LAS↑	Acc.↑	LAS↑
w/o Rationales	71.44	-	53.76	-	66.40	-	53.32	-
Standard Training	73.98	29.50	62.78	2.29	69.9	26.04	53.64	50.76
Dropout Context	71.81	**30.13**	60.27	2.58	68.6	25.06	53.48	51.55
PINTO	**74.73**	29.36	**63.33**	**3.66**	**72.6**	**26.72**	**56.32**	51.19

A.2 Ablation on the LM size for the Rationalizing Module

Table 8 shows the results of the Pipeline approaches using LMs with different model sizes as the rationalizing module.

Table 7: **OOD Results.** Performance (accuracy) of the compared methods using RoBERTa-Large as the reasoning module, which are firstly fine-tuned on a source dataset and then directly predict on a target dataset (denoted as $source \rightarrow target$).

Method	**CSQA→OBQA**	**CSQA→QASC**	**OBQA→CSQA**	**QASC→CSQA**	**QASC→OBQA**
w/o Rationales	49.55	53.67	44.51	60.48	54.15
Standard Training	54.60	55.75	46.62	60.26	55.95
Dropout Context	52.55	55.40	41.01	58.60	58.90
PINTO	54.30	**56.83**	**46.95**	**61.61**	**60.95**

Table 8: **Ablation on the LM with different model sizes as the rationalizing module.** We bold the results that outperform the second-best method with statistical significance ($p < 0.05$).

	GPT-2 (1.5B)		**GPT-J (6B)**		**GPT-neox (20B)**		**GPT-3 (175B)**	
Method	Acc.↑	LAS↑	Acc.↑	LAS↑	Acc.↑	LAS↑	Acc.↑	LAS↑
Standard Training	59.19	13.64	59.44	15.81	59.48	18.75	59.40	22.99
Dropout Context	58.82	13.33	58.48	15.70	59.64	20.40	58.64	23.51
PINTO	**61.10**	**14.63**	**60.73**	**16.09**	**61.67**	**24.22**	**61.32**	**23.54**

A.3 Variance statistics of all the fine-tuned models

Table 9 and Table 10 show the variance statistics (standard deviation) along with ID and OOD task performance (accuracy) of the fine-tuned methods from Table 2 and Table 3.

Table 9: **ID Results with variance statistics**. Task performance (accuracy) and variance statistics (standard deviation) of the fine-tuned methods from Table 2.

	CSQA		**StrategyQA**		**OpenBookQA**		**QASC**	
Method	Acc.↑	STD	Acc.↑	STD	Acc.↑	STD	Acc.↑	STD
w/o Rationales	58.68	0.19	58.12	2.53	55.85	0.90	35.58	0.64
Distill. Self-Ra. (Explain-First)	51.97	0.41	50.20	4.03	48.90	0.78	33.34	1.18
Distill. Self-Ra. (Predict-First)	55.77	0.69	54.61	1.87	50.25	3.45	34.53	0.70
Standard Training	59.48	0.41	57.11	2.66	56.65	0.70	37.50	0.55
Dropout Context	59.64	0.89	51.45	2.76	57.55	1.08	35.37	1.87
PINTO	61.67	0.31	60.87	2.66	58.85	1.37	37.82	1.48

A.4 Human evaluation on the rationales

We conducted a human evaluation of 100 generated rationales from the CSQA dataset. The evaluation is a head-to-head comparison between the human-annotated rationales and the machine-generated rationales. Annotators were asked to judge for 5 dimensions on a 3-point Likert scale following a prior work (Wiegreffe et al., 2021): 1) Factuality (How factual is this Explanation?) 2) Grammaticality (Is this Explanation grammatical?) 3) New Info (Does the Explanation provide new facts, information, or reasoning not stated in the Question and Answer?) 4) Supports Answer (Is the Explanation relevant to justifying the Answer?) 5) Completeness (Does the Explanation provide enough information to jusify the answer?)

We obtain a fair level of agreement measured by Fleiss Kappa (k=0.34) for the evaluation. The results in Table 11 show that machine-generated results are competitive with human annotation on most of the evaluating dimensions. Generated rationales are even judged to be more grammatical than human annotations. As for completeness, generated rationales are slightly worse than human annotations. We think this is because the human annotators were explicitly encouraged to provide more comprehensive rationales when annotating the CSQA dataset (Aggarwal et al., 2021).

Table 10: **OOD Results with variance statistics**. Task performance (accuracy) and variance statistics (standard deviation) of the fine-tuned methods from Table 3.

	CS→OB		**CS→QASC**		**OB→CS**		**QASC→CS**		**QASC→OB**	
Method	Acc.↑	STD	Acc.↑	STD	Acc.↑	STD	Acc.↑	STD	Acc.↑	STD
w/o Rationales	32.05	0.90	39.17	1.54	24.87	1.52	45.74	1.19	34.90	2.06
Distill. Self-Ra. (Explain-First)	24.85	1.51	31.43	0.85	23.05	1.42	43.16	1.00	31.55	0.71
Distill. Self-Ra. (Predict-First)	25.10	1.40	32.26	0.64	26.43	1.42	45.17	0.88	30.50	0.87
Standard Training	31.05	1.10	40.04	1.13	25.37	0.7	47.71	1.15	34.50	1.46
Dropout Context	32.30	1.57	38.85	0.70	23.01	1.14	44.27	2.36	32.90	2.06
PINTO	34.90	1.37	42.25	0.35	27.66	1.86	48.03	0.97	35.75	1.47

Table 11: **Human evaluation on the human-annotated and machine-generated rationales for CSQA.** A fair level of agreement measured by Fleiss Kappa (k=0.34) is obtained among the annotators.

Rationale Source	Factuality	Grammaticality	New Info	Supports Answer	Completeness
Human	0.94	0.98	0.71	0.90	0.82
Generated	0.91	0.99	0.69	0.87	0.65