STaR-plus: building robust and efficient language model reasoners

Stanford CS224N Custom Project

Kunal Sinha (Author)
Department of Symbolic Systems
Stanford University
ksinha2@stanford.edu

Jesse Mu (Mentor)
Department of Computer Science
Stanford University
muj@stanford.edu

Abstract

Generating step-by-step "chain of thought" rationales has been shown to improve the performance of Large Language Models (LLMs) on commonsense reasoning tasks. As such, recent algorithms such as the Self-Taught Reasoner (STaR) have been proposed to encourage high-quality rationale generation from a limited number of examples. However, it is unclear whether LLMs display two abilities that allow humans to reason effectively: first, the ability to apply insights from simpler problems to solve more challenging out-of-distribution problems, and second, the ability to learn heuristics that speed up the reasoning process. We demonstrate that LLMs can do both. Specifically, we first show that the existing STaR algorithm allows for effective out-of-distribution generalization. Second, we present STaR-plus, a variant of the original algorithm that achieves comparable accuracy despite employing shorter rationales on average.

1 Introduction

Large Language Models (LLMs) have recently seen dramatic improvements in performance on question-answering tasks as a result of Chain-of-Thought prompting. This technique involves prompting models to explicitly generate intermediate reasoning steps before generating the final answer. Nye et al. (2021) find that the use of such "scratchpads" yields strong performance on arithmetic problems, and Shwartz et al. (2020) find that the approach also succeeds on natural language problems that require commonsense reasoning.

Unfortunately, directly finetuning the model to produce such rationales is often infeasible due to logistical constraints. Designing a training dataset of question-answer pairs with annotated rationales is expensive and time-consuming, given that human volunteers must supply these rationales manually. Generating rationales automatically based on templates is not feasible in domains where multiple valid rationales can lead to the same answer (e.g in most natural language domains). Furthermore, models that attempt to avoid the need for finetuning by employing few-shot learning typically underperform their finetuned counterparts (Nye et al., 2021).

In response to these challenges, Zelikman et al. (2022) introduce the Self-Taught Reasoner (STaR) algorithm. Given a small number of example question-answer pairs with annotated rationales, STaR bootstraps the ability to perform high quality rationale generations. Specifically, the algorithm few-shot prompts the model to produce rationales, filters out generations that yield incorrect final answers, and uses the remaining generations as a new training set to finetune on. This process repeats for a fixed number of iterations.

This simple iterative algorithm was shown to drastically improve performance on a variety of tasks such as arithmetic, math word problems, and commonsense question-answering. For example, in the CommonSenseQA dataset (Talmor et al., 2018), a model trained according to STaR achieved higher accuracy than models relying on few-shot prompting ($+35.9\%$) or direct finetuning ($+12.5\%$).

Furthermore, the STaR model achieved an accuracy within 0.5% of a finetuned model that was 30 times larger.

In this paper, we seek to investigate and further improve two dimensions of STaR's reasoning capabilities: the robustness of the model's performance on out-of-distribution data, and the efficiency of the reasoning employed by the model. We accordingly split our work into two tasks.

The first task examines the model in a setting where primarily easy problems are seen during training time, but the model must answer more difficult ones during test time. This task carries significance for a few reasons. First, this setting more accurately resembles a real world setting where the distribution of problems is skewed. In most domains, a dataset of problems will likely include a very few number of difficult problems, because these are more time-consuming to generate and can be found less frequently in a corpus. Second, in most applications of language models, the model may encounter high variance in the difficulty of questions asked by users; model performance should be robust to this variance.

The second task introduces STaR-Plus, a variant of the STaR algorithm that aims to promote shorter rationales on average, without sacrificing accuracy. This efficiency in reasoning is similarly valuable in most user-facing applications of language modeling. Indeed, rationale generation can serve as a bottleneck that increases the time elapsed between question and answer, which may negate the gains in accuracy from a user experience perspective; shrinking this bottleneck therefore strengthens the appeal of STaR. Additionally, encouraging shorter rationales may teach the model to learn shortcuts or heuristics that can help solve complex problems without reasoning from first principles (which could quickly become unwieldly). Notably, this behavior would mimic that of human reasoners (Karlan, 2021).

2 Related Work

In-context learning Few-shot prompting in language models involves providing examples of high-quality problem-solution pairs before a presenting a new problem. This simple technique has allowed LLMs to achieve high performance on a variety of specific tasks (e.g cloze tasks, translation, commonsense reasoning) that matches that of previous state-of-the-art models. Crucially, LLMs can learn to solve new tasks without any explicit finetuning or gradient updates (Brown et al., 2020). Previous work has sought to explain this in-context learning ability by suggesting LLMs implicitly perform Bayesian Inference (Xie et al., 2020). Other authors elucidate the phenomenon at a lower level of abstractions, discussing the role of "induction" attention heads that learn simple algorithms for pattern recognition in tokens (Olsson et al., 2020).

Rationales Rajani et al. (2019) first introduce the prospect of language models generating intermediate rationales to improve performance on commonsense reasoning tasks. They manually craft the Common-Sense-Explanations dataset (CoS-E) and find that an LLM finetuned on this dataset achieves performance 10% above state-of-the-art models. Nye et al. (2021) extend upon this work by demonstrating that transformers can be trained to produce "scratchpads" that contain the results to intermediate computations when solving math problems. Some authors have sought to provide specific theoretical explanations for the benefits of rationale generation. For example, Zhou et al. (2020) describe natural language rationales as latent variables modeling the underlying reasoning processes of the LLM. Furthermore, Wies et al. (2023) formally show that certain problems that are unlearnable by an LLM can be decomposed into a polynomial number of learnable subproblems, each of which depends on $O(1)$ previous subproblems. Asking language models to produce rationales therefore encourages this breakdown and thereby improves answer accuracy.

3 Approach

3.1 STaR algorithm

As explained earlier, STaR iteratively leverages a small number of labeled rationales to bootstrap effective rationale generation. Intuitively, this process involves repeatedly finetuning the language model on its own generated rationales that are "high quality." Zelikman et al. (2022) define a high-quality rationale as one that leads to the correct answer, and find that this simple heuristic is sufficient to achieve good performance. We describe the algorithm below in greater detail.

As input, the algorithm takes in pretrained LLM M and a dataset of question-answer pairs $\mathcal{D} = \{x_i, y_i\}|_{i=1}^{D}$ where x_i is a question and y_i is its associated answer. Additionally, we supply a much smaller dataset with labeled rationales $\mathcal{P} = \{x_i, r_i, y_i\}|_{i=1}^{P}$, where r_i is a rationale. We then take the following steps:

1. For each question $x_i \in D$: have the model M generate predictions for rationales $\hat{r}_i$ and the final answer $\hat{y}_i$. Do so by performing few shot prompting with the examples in $\mathcal{P}$. We can call the list of outputs $\mathcal{O}$.
2. Create a new list $\mathcal{O}'$ by filtering out any prediction $\{x_i, \hat{r}_i, \hat{y}_i\} \in \mathcal{O}$ where $\hat{y}_i \neq y_i$, meaning the model generated an incorrect answer.
3. Finetune M on this filtered list $\mathcal{O}'$ until convergence
4. Repeat steps 1-3 for some fixed number of N iterations.

Observe that the algorithm contains inner loop (steps 1-3) that finetunes the model M over multiple epochs, and an outer loop (step 4) that repeats the process N times.

This algorithm's inner loop can be modeled as a reinforcement learning problem: for each question $x_i \in \mathcal{D}$, we sample rationales $\hat{r}_i$ and answers $\hat{y}_i$ from our model M, which represents sampling a particular trajectory from our current policy. Filtering out predictions with incorrect answers represents maximizing the indicator reward function $\mathbb{1}[\hat{y}_i = y_i]$. Therefore, STaR maximizes the following objective function, where the gradient is derived through the standard log-deriviative trick:

$$J(M, X, Y) = \sum_i \mathbb{E}_{\hat{r}_i, \hat{y}_i \sim p_M(\cdot|x_i)}(\mathbb{1}[\hat{y}_i = y_i]) \tag{1}$$

$$\nabla J(M, X, Y) = \sum_i \mathbb{E}_{\hat{r}_i, \hat{y}_i \sim p_M(\cdot|x_i)}(\mathbb{1}[\hat{y}_i = y_i] \cdot \log P_M(\hat{y}_i, \hat{r}_i|x_i)) \tag{2}$$

For the first task of testing out-of-distribution performance, we keep the underlying STaR algorithm and its objective function unchanged; this ensures that we isolate the effects of manipulating the data distribution.

3.2 STaR-Plus

Observe that the existing STaR algorithm does not penalize the model for excessively long rationale generations. As such, for the second task of encouraging efficient reasoning, we introduce STaR-Plus, a variant of the original with an altered objective function.

In the original objective function J, the use of the indicator reward $\mathbb{1}[\hat{y}_i = y_i]$ ensures that training examples with correct answers exert equal influence on the gradient and those with incorrect answers exert none. However, the STaR-Plus objective J' applies a length penalty to the reward function. Examples with correct answers now exert an influence that is inversely proportional to their length, promoting brevity in rationale generation. We control the strength of the penalty with a hyperparameter α.

$$J'(M, X, Y) = \sum_i \mathbb{E}_{\hat{r}_i, \hat{y}_i \sim p_M(\cdot|x_i)}(-\alpha|\hat{r}_i| \cdot \mathbb{1}[\hat{y}_i = y_i]) \tag{3}$$

$$\nabla J'(M, X, Y) = \sum_i \mathbb{E}_{\hat{r}_i, \hat{y}_i \sim p_M(\cdot|x_i)}(-\alpha|\hat{r}_i| \cdot \mathbb{1}[\hat{y}_i = y_i] \cdot \log P_M(\hat{y}_i, \hat{r}_i|x_i)) \tag{4}$$

Increasing the value of α produces a stronger penalty for large rationales. However, an excessively high value could yield large gradient updates that increase the risk of the loss curve overshooting the optimum. As such, we conduct hyperparameter tuning as described in section 4.3.

As a further caveat, note that the distribution of generated rationale lengths often has a high variance. The length penalty can therefore produce high variance in the magnitude of each gradient step, yielding instability during training and also potentially lowering the chances of convergence. To avoid this risk, we apply z-score normalization to the length distribution. This normalization also has additional benefits: centering the distribution at 0 ensures that the normalized $|\hat{r}_i|$ in equation 4 will be positive for examples with above-average-length rationales and negative for those with below-average-length. As such, we effectively minimize the objective on the former and maximize the objective on the latter during training. Z-score normalization therefore assists in encouraging brevity.

4 Experiments

4.1 Data

We use the String Editing dataset developed by Andreas et al. (2019). This dataset consists of a series of communication games involving two agents: a speaker and a listener. The speaker provides multiple examples of text undergoing modification according to an implicit rule. For example, given the rule "replace vowels with mx", a speaker might provide $editor \rightarrow mxdmxtmxr$ and $being \rightarrow bmxmxng$. The speaker then offers a new string of text, e.g $lobbied$. The listener must infer the modification rule from the previous examples and apply that rule to produce a new output. (In this case, the correct output is $lmxbbmxmxd$).

We preprocess the data to adhere to the format required by the STaR algorithm. For each problem, we define x_i as the input-to-output mappings provided by the speaker as well as the new input, and y_i as the new output. We define the rationale $\hat{r}_i$ as an explanation of the modification rule in natural language. We provide an example below.

```
x_i:
e d i t o r → m x d m x t m x r
t o n e s → t m x n m x s
e x c e p t i o n → m x x c m x p t m x m x n
b e i n g → b m x m x n g
l o b b i e d → ?
r_i: replace vowels with m x
y_i: l m x b b m x m x d
```

Because our first task involves testing out-of-distribution generalization, we perform further preprocessing to stratify the data by difficulty. For a given problem, the length of the groundtruth rationale r_i serves as a suitable proxy for difficulty. Accordingly, we sort the array of problems with respect to groundtruth rationale length and split the array into $K = 10$ buckets of uniform size. Because the average length increases in each consecutive bucket, we can treat each bucket as representing a "difficulty ranking." The last few buckets contain the most difficult problems.

The dataset contains 4,000 total examples, with each natural language rationale supplied manually by human volunteers. (The authors allocate 3,000 to the training set and the remaining 1,000 to the validation and test sets). The relatively small size of this dataset presents a promising opportunity to showcase the strengths of STaR; the bootstrapping framework is designed to boost accuracy substantially from a small number of high-quality annotated examples. Furthermore, unlike other datasets such as CommonSenseQA (Talmor et al., 2018), the String Editing dataset does not require prior world knowledge to answer the questions correctly. As such, experiments with this dataset can examine the reasoning capabilities of an LLM in isolation, specifically the capacity for pattern recognition.

In the following experiments, we split the dataset into four groups: pretrain, self-play, validation, and test. The pretrain dataset refers to $\mathcal{D}$ described in section 3.1, whereas the selfplay dataset refers to $\mathcal{P}$. We set the size of the selfplay dataset to be significantly lower than that of the pretrain, to more accurately simulate a real-world setting.

4.2 Evaluation method

For both tasks, we use the following metrics:

- Test accuracy: the percent of answers predicted by the model $\hat{y}_i$ that have an exact string match with the groundtruth answer y_i.

- Average edit distance: the average of the edit distances between each pair $(\hat{y}_i, y_i)$ of predicted answers and groundtruth answers in the dataset. We define edit distance as the Levenshtein distance, i.e the number of single-character insertions, deletions, or substitutions needed to transform $\hat{y}_i$ to y_i.
- Validity: the percent of model outputs that follow the expected format
- Rationale accuracy: the percent of rationales generated by the model $\hat{r}_i$ that match the groundtruth rationale r_i.

The test accuracy serves as the primary metric to assess model performance. However, because test accuracy relies on an exact string match between $\hat{y}_i$ and y_i to classify a prediction as accurate, the metric will fail to capture subtler improvements in generation quality where the prediction simply moves closer to the groundtruth. The average edit distance metric exists to capture this type of improvement by imposing a softer standard for accuracy.

The validity metric exists primarily as a sanity check. We expect validity to plateau at approximately 100%; a low validity would suggest that the model did not understand the task provided, and would signal the need for more examples in few-shot prompt.

Rationale accuracy provides information on the quality of the reasoning employed by the model. However, given that the same underlying string editing rule can be expressed in multiple unique ways in natural language, even a model with high test accuracy could perform badly on this metric. This metric thus exists primarily to examine how closely the model's reasoning will mimic that of the human volunteers who supplied the annotations in the dataset.

For the second task, we introduce two new metrics.

- Average predicted rationale length: the average length of $\hat{r}_i$ across all examples $i \in D$.
- Average groundtruth rationale length: the average length of r_i across all examples $i \in D$.

A decrease in the first average suggests an improvement in the efficiency of the reasoning employed by the language model. The second average represents the gold standard of efficiency that human reasoners are capable of, and therefore represents an upper bound for the first average. (An increase beyond this upper bound would suggest that the language model has discovered heuristics that human reasoners do not readily employ).

In addition to the averages, we also plot the length distributions for the predicted and groundtruth rationales. These graphs simply provide more information in how the averages are affected by skew or potential outliers.

4.3 Experimental details

For our language model, we use the smallest version of GPT-2 with 124 million parameters (Radford et al., 2019). Our experiments have two sets of relevant hyperparameters: those pertaining to the inner and outer loop of STaR respectively (as defined in section 3.1). In the inner loop, we finetune GPT-2 for 25 epochs with an early stopping patience of 4 epochs, applying Adam Optimization for gradient updates. We use a fixed learning rate of 0.0001 and a batch size of 4. In the outer loop, we set the number of iterations to 5. For the other hyperparameters, we follow the configuration described in Zelikman et al. (2022).

The first few experiments establish a baseline. We first train the model directly on 3,000 examples without initiating the the STaR algorithm. These results represent an idealized scenario where we have access to a large question-answer dataset with annotated rationales and therefore do not need to apply STaR's bootstrapping approach. The test accuracy in this experiment signifies an upper bound for the following experiments.

Next, we run an experiment where we assign 2,000 examples to the pretrain dataset and the other 1,000 to the selfplay dataset. This experiment represents a more realistic scenario where we have access to fewer examples with annotated rationales. The goal is to test whether STaR can push accuracy towards the idealized upper bound despite the paucity of high-quality data.

After verifying that STaR boosts performance in the baseline setting, we begin our first task: investigating whether the algorithm allows for out-of-distribution generalization on more difficult problems.

We run an experiment where the contents of difficulty buckets 1-6 are assigned to the pretrain set and the contents of 7-10 are assigned to the selfplay. (We form these buckets as described in section 4.1). This experiment represents the scenario where we have access to a large number of easy problems but a smaller number of difficult ones.

Next, we perform our second task: encouraging brevity in rationale generation. We revert to the earlier baseline setting with 2,000 pretrain examples and 1,000 selfplay examples drawn from the same distribution. However, we employ the STaR-Plus algorithm with a length penalty of $\alpha = .01$. We select the value of α through a hyperparameter search across multiples of 10^{-i} for i from 1 to 4.

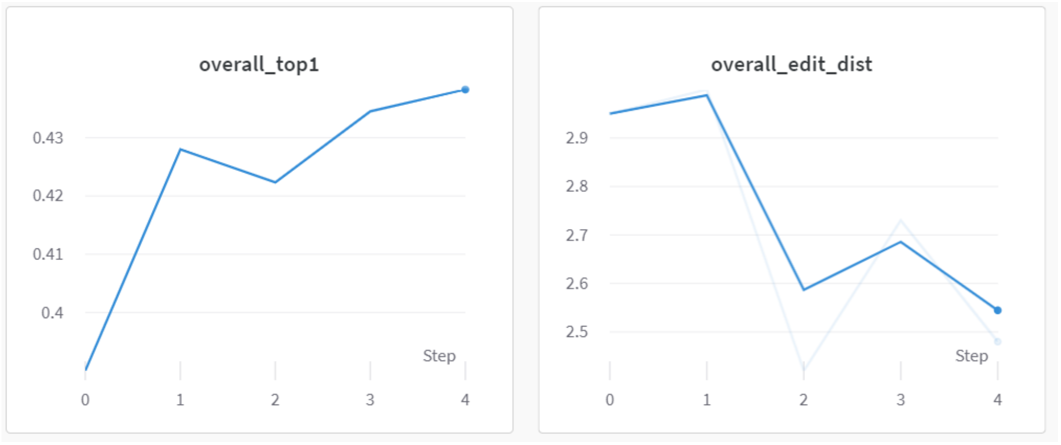
4.4 Results

4.4.1 Baseline

When training the model directly on 3,000 examples, we obtain the following measurements. Recall that this experiment represents an idealized setting and therefore provides a set of upper bounds for each metric in future experiments.

- Test accuracy: 43.23%
- Average edit distance: 2.1
- Validity: 94.9%
- Rationale accuracy: 50%

We now plot the metrics for the second baseline experiment, where we take 2,000 examples as pretrain data and the other 1,000 and selfplay data. Because we now apply the STaR algorithm, we graph these metrics across all 5 iterations of the outer loop.



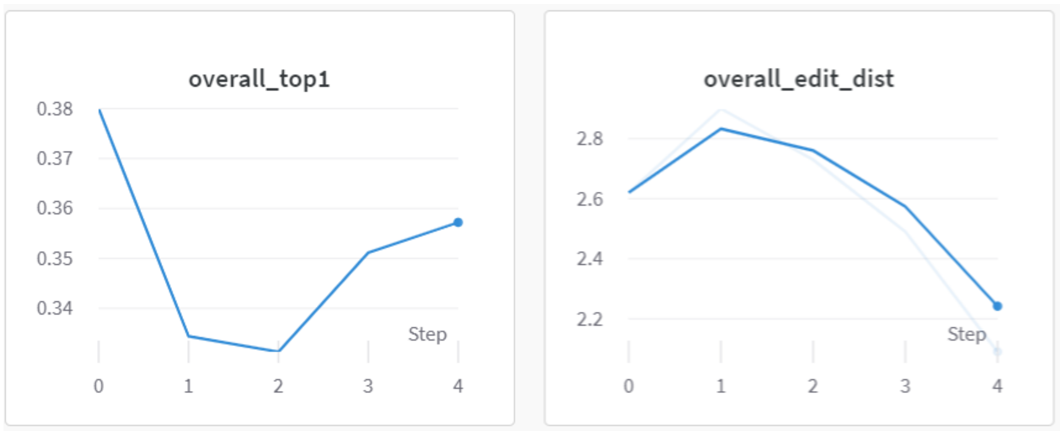
These two metrics correspond to test accuracy and average edit distance respectively. (For the sake of space, we move all graphs for validity and rationale accuracy to appendix A. These metrics provide useful additional information but do not directly capture answer accuracy, the primary meeasure of interest).

Observe that although the test accuracy begins at 38%, the accuracy rises consistently across each iteration of the outer loop until touching the gold standard of 43%. Although the increase is modest, the result demonstrates the ability of the STaR algorithm to achieve comparable accuracy to the gold standard of the finetuned model, despite having a fraction of the rationale-annotated examples. We similarly find that the average edit distance drops from 2.95 to 2.54, demonstrating that the model predictions gradually grow closer to the groundtruth. However, they still lag slightly behind the gold standard of 2.1, suggesting the need for further adjustments to the training process (e.g more hyperparameter tuning).

Finally, we obtain an average predicted rationale length of 125.34 compared to an average groundtruth rationale length of 124.45. We discuss these results further in section 4.4.3.

4.4.2 Task 1: Robustness

Recall that in the out-of-distribution tests, we supply the first 6 buckets as pretrain data and the latter 4 as selfplay data. The relevant metrics are shown below

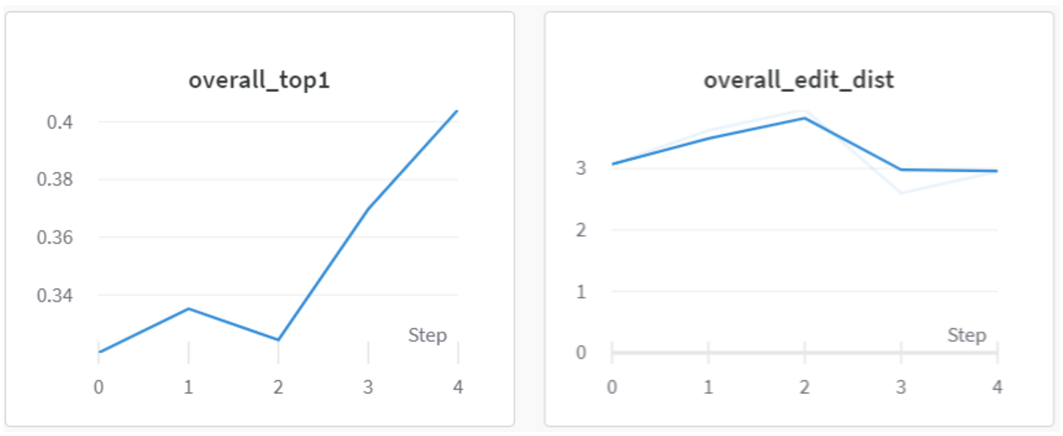


After the first iteration of the outer loop, we observe a steep drop in test accuracy from 38% to 33.4%. However, accuracy does partially recover in subsequent iterations, rising to 35.7%. The accuracy overall stays below the 38% − 43% range observed in the baseline. The relatively weaker performance is expected given the fact that we expose the model to data in test time that was unseen during pretraining. However, the drop in accuracy initially seems to suggest that STaR does not succeed in out-of-distribution generalization.

However, the average edit distance metric yields a different interpretation. While rising slightly from 2.62 to 2.83 after the first iteration, the distance eventually drops to 2.242, moving closer towards the gold standard of 2.1. These results suggest that STaR actually does improve performance in gradual manner. We discuss hypotheses for why the two metrics contradict one another in section 5.

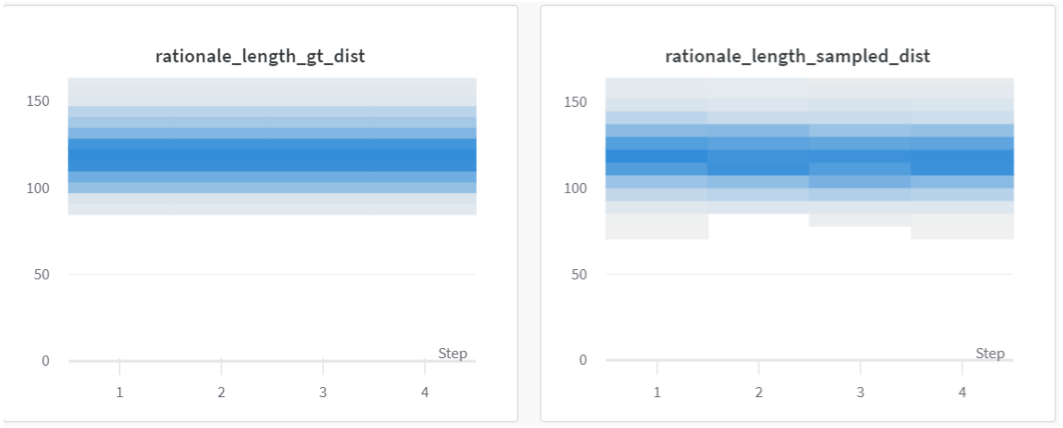
4.4.3 Task 2: Efficiency

We begin presenting the same two accuracy metrics for the experiment with STaR-Plus. Recall that we reset to baseline settings while optimizing for the modified objective function.



Observe that the test accuracy starts at a low value of 32% but eventually rises to 40.4%, only slightly lower than the gold standard of 43.2%. However, the average edit distance remains mostly flat around 3, which is higher than in the baseline. These results suggest that STaR-Plus does suffer from some drops in accuracy, which is expected given the decision to prioritize succinct over descriptive rationales; however, the drop is not drastic.

We also graph the distribution of groundtruth rationale lengths compared to the generated rationale lengths.

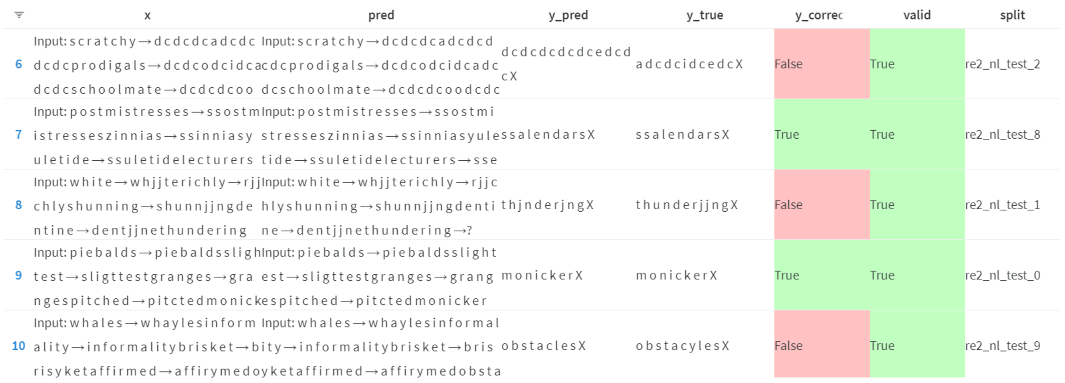


The average groundtruth rationale length is the same as in the baseline: 124.45. However, the average predicted rationale length is now 123.8 (compared to 125.34). These results suggest that the length penalty does encourage shorter rationales, but only very modestly. More hyperparameter tuning with α is likely required; we discuss the significance of these results further in section 5.

5 Analysis

The results described above raise a few questions that merit further discussion. First, in the baseline experiment, the gold standard for accuracy seems fairly low at 43.23%. However, note that we define an output as accurate only if every character in the string matches the groundtruth. The chance performance of such a system shrinks exponentially as the length of the groundtruth string increases, quickly approaching 0%. As such, accuracy close to 50% suggests the model does learn the required behavior to a reasonable degree. Further considerations include the fact that we use the smallest version of GPT-2 for these experiments and use a dataset with only 3,000 examples. These choices partially reflect logistical constraints of the project; future work should train larger models on more extensive datasets.

For Task 1, we observe in section 4.4.2 that the test accuracy and the edit distance metrics yield conflicting interpretations; the former suggests a drop in accuracy while the latter suggests a gradual improvement. We hypothesize that test accuracy is less resistant to noise. Specifically, given the paucity of training data, the chances of the model sampling the correct output are similar to those of sampling a very close yet incorrect output (e.g with an edit distance of 1). Therefore the test accuracy provides a less reliable gauge of model performance overall. In particular, the metric masks any more gradual improvements that fall below the threshold of perfect accuracy. We can observe this phenomenon by examining a set of generated rationales randomly selected from this experiment.

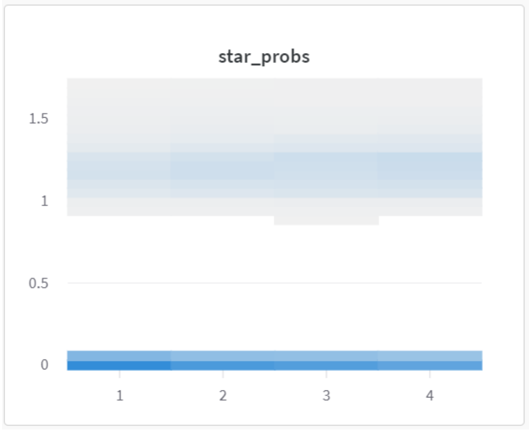


	x	pred	y_pred	y_true	y_correc	valid	split
6	Input: scratchy→dcdcdcadcdc dcdcprodigals→dcdcodcidca dcdcschoolmate→dcdcdcoo	Input: scratchy→dcdcdcadcdcd cdcprodigals→dcdcodcidcadc dcschoolmate→dcdcdcoodcdc	dcdcdcdcdcedcd cX	adcdcidcedcX	False	True	re2_nl_test_2
7	Input: postmistresses→ssostm istresseszinnias→ssinniasy uletide→ssuletidelecturers	Input: postmistresses→ssostmi stresseszinnias→ssinniasyule tide→ssuletidelecturers→sse	ssalendarsX	ssalendarsX	True	True	re2_nl_test_8
8	Input: white→whjjterichly→rjj chlyshunning→shunnjjngde ntine→dentjjnethundering	Input: white→whjjterichly→rjjc hlyshunning→shunnjjngdenti ne→dentjjnethundering→?	thjnderjngX	thunderjjngX	False	True	re2_nl_test_1
9	Input: piebalds→piebaldsligh test→sligttestgranges→gra ngespitched→pitctedmonicke	Input: piebalds→piebaldsslight est→sligttestgranges→grang spitched→pitctedmonicker	monickerX	monickerX	True	True	re2_nl_test_0
10	Input: whales→whaylesinform ality→informalitybrisket→b risyketaffirmed→affirymedo	Input: whales→whaylesinformal ity→informalitybrisket→bris yketaffirmed→affirymedobsta	obstaclesX	obstacylesX	False	True	re2_nl_test_9

In the table, x and $pred$ represent the full groundtruth example and the full prediction respectively. The two adjacent columns, y_pred and $y\_correct$, explicitly contrast the predicted final output to the groundtruth. Among these examples, the model has an approximately equal chance of guessing correctly or incorrect. However, many of the incorrect examples have very low edit distance (consider $obstayclesX$ versus $obstaclesX$).

Regarding Task 2, we find that although the STaR-Plus does encourage gains in efficiency, these gains are fairly modest, reducing the rationale by only a few tokens on average. One possible explanation for this finding is that the baseline STaR algorithm already exhibited an average predicted rationale length that was close to the groundtruth (fewer than 2 tokens apart). If the groundtruth rationales provided by human reasoners represent an upper bound on efficiency, then STaR-Plus cannot provide much further improvement. The high degree of efficiency in the baseline could suggest that the basic STaR algorithm is sufficient to encourage brevity in rationales despite the lack of an explicit length penalty. In contrast, the baseline could have achieved such results simply due to insufficient variance in the dataset's groundtruth rationale lengths; the lack of exposure to long rationales during training could have prevented the risk of the model learning to generate long sequences.

On a related note, future work must also investigate whether the groundtruth rationales provided by human reasoners even represent the gold standard in the first place. A model could theoretically learn to produce significantly shorter rationales by leveraging heuristics that humans tend not to. Answering this question requires performing further hyperparameter tuning across values of α. We graph the effects of our current choice of $\alpha = 0.01$ on the objective function J' below.



Here we display a histogram of the values of the term inside the derivative of our objective function $\nabla J'$, namely $(-\alpha|\hat{r}_i| \cdot \mathbb{1}[\hat{y}_i = y_i] \cdot \log P_M(\hat{y}_i, \hat{r}_i|x_i))$, as decribed in equation 4. As expected, we have a cluster of points at 0. These represent examples where the model generated an incorrect final answer prediction $\hat{y}_i$. However, for the examples where the model generated a correct predictions, the values spread from roughly 0.9 to 1.75. Larger values represent examples with shorter rationales that had a stronger effect on the gradient. It is possible, however, that the magnitude of these values must increase beyond 1.75 to obtain a stronger length penalty effect. Ensuring that we can do this without creating instability during training, as discussed in section 3.2, is an open issue that requires further investigation.

6 Conclusion

We find that STaR is able to effectively bootstrap the ability for high-quality rationale generation from a small subset of the original dataset, achieving similar results to a model directly finetuned on the entire dataset. Critically, the algorithm displays the ability to generalize reasonably well to difficult out-of-distribution problems. The introduction of the novel STaR-Plus approach allows for modest gains in efficiency with regards to reasoning, but suffers some drops in accuracy compared to the original STaR. Future work should seek to expand upon STaR-Plus, experimenting with alternate objective functions that more effectively balance rationale brevity with answer accuracy.

References

Jacob Andreas, Dan Klein, and Sergey Levin. 2019. Learning with latent language. In *Neural and Evolutionary Computing*.

Tom Brown, Benjamin Mann, and et al. Ryder, Nick. 2020. Language models are few-shot learners. In *Advances in neural information processing systems*.

Brett Karlan. 2021. Reasoning with heuristics. In *Ratio*.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, and David Luan. 2021. Show your work: Scratchpads for intermediate computation with language models. In *Neural and Evolutionary Computing*.

Catherine Olsson, Nelson Elhage, and Neel et al. Nanda. 2020. In-context learning and induction heads. In *Transformer Circuits Thread*.

Alec Radford, Jeffrey Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. 2019. Language models are unsupervised multitask learners. In *OpenAI blog*.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Association of Computational Linguistics (ACL)*.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *North American Chapter for the Association of Computational Linguistics (NAACL)*.

Noam Wies, Yoav Levine, and Amnon Shashua. 2023. Sub-task decomposition enables learning in sequence to sequence tasks.

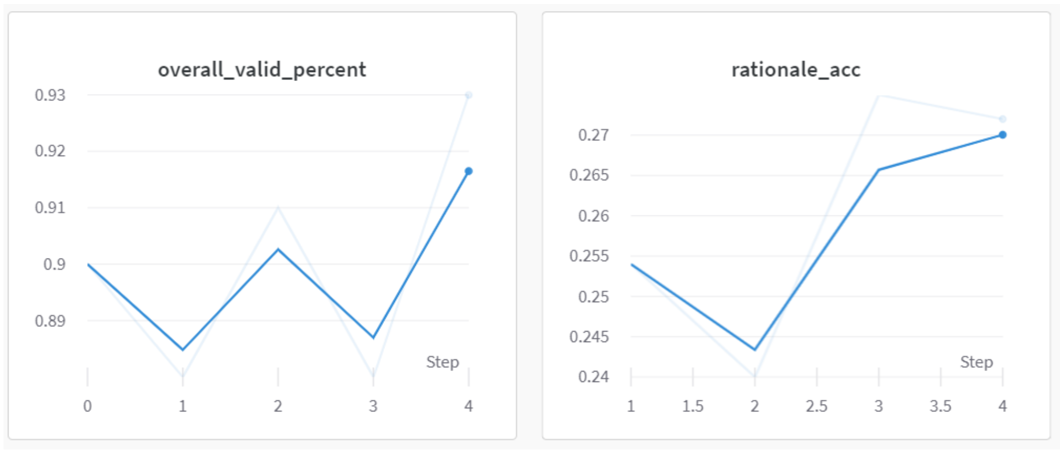
Michael Sang Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2020. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations (ICLR)*.

Eric Zelikman, Jesse Mu, and Noah Goodman. 2022. Star: Self-taught reasoner, bootstrapping reasoning with reasoning. In *Conference on Neural Information Processing Systems (NeurIPS)*.

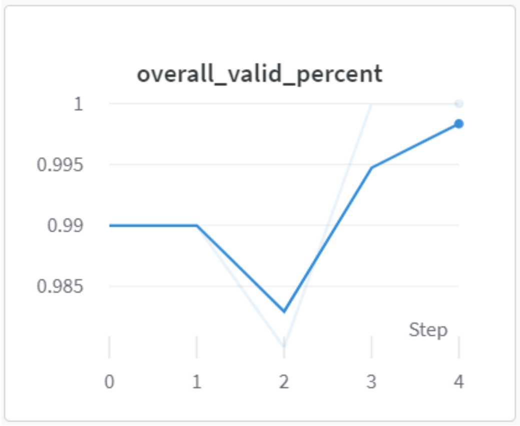
Wangchunshu Zhou, Jinyi Hu, Hanlin Zhang, and Xiaodan et al Liang. 2020. Towards interpretable natural language understanding with explanations as latent variables. In *Advances in Neural Information Processing Systems*, volume 33, pages 6803–6814. Curran Associates, Inc.

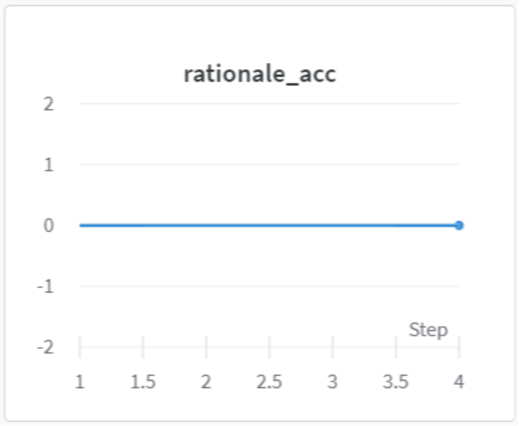
A Appendix

A.1 Baseline (additional graphs)

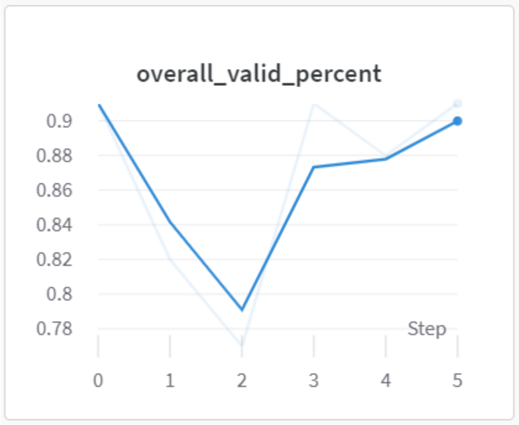


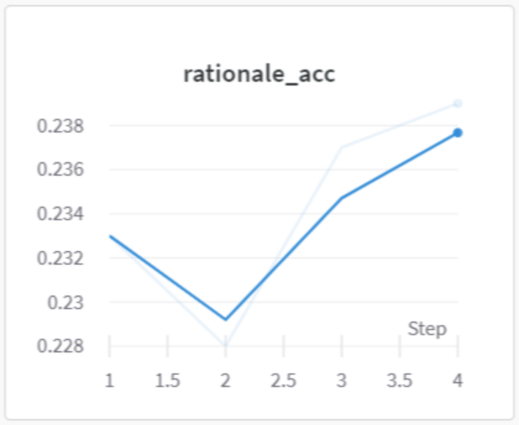
A.2 Task 1 (additional graphs)





A.3 Task 2 (additional graphs)


overall_valid_percent
Step


rationale_acc
Step