Teaching Models to Improve on Tape

Liat Bezalel[1], **Eyal Orgad**[1], **Amir Globerson**[1,2]

[1]Tel Aviv University
[2]Google Research
liatbezalel@mail.tau.ac.il, eyalorgad@mail.tau.ac.il, gamir@tauex.tau.ac.il

Abstract

Large Language Models (LLMs) often struggle when prompted to generate content under specific constraints. However, in such cases it is often easy to check whether these constraints are satisfied or violated. Recent works have shown that LLMs can benefit from such "corrective feedback". Here we claim that this skill of LLMs can be significantly enhanced via training. We introduce an RL framework for teaching models to use such rewards, by simulating interaction sessions, and rewarding the model according to its ability to satisfy the constraints. We refer to our method as CORGI (Controlled Generation with RL for Guided Interaction), and evaluate it on a variety of controlled generation tasks. We find that CORGI consistently outperforms the baseline reinforcement learning method that does not incorporate conversational feedback. Furthermore, CORGI's interactive framework enables meta-learning, allowing the LLM to better generalize to guided interaction in new tasks. Our results clearly show that conversational optimization, when combined with reinforcement learning, significantly improves the effectiveness of LLMs in controlled generation contexts.

Code — https://github.com/Liat-Bezalel/corgi

Introduction

Large Language Models (LLMs) have become quite effective at following instructions from users. Despite this progress there are still many tasks where LLMs will not respond correctly. A particular subclass of cases is those where it is relatively easy to judge whether a response is correct or not. For example, if the task is to generate a sentence with six words, it is fairly easy for to check if this constraint is violated.

A natural question is whether such feedback can be used in order to make the LLM produce a correct response. This exciting prospect has been considered in several works recently, including Madaan et al. (2023) and Shinn et al. (2024). Indeed, these works show that feedback can be "written to the LLM context", and the LLM can use this to improve its generation. Furthermore, it has been shown that LLMs can learn to utilize external tools (Schick et al. 2023; Lu et al. 2023), thereby improving their accuracy.

Building on the potential of feedback, we aim to explore whether LLMs can use this signal more effectively. We focus on controlled generation tasks, where LLMs often struggle (Sun et al. 2023). Specifically, we consider the important subclass of tasks where it is easy to check if the generated text satisfies the desired control requirements (e.g., the "generate sentence with six words" example above). We hypothesize that LLMs can be trained to more effectively use corrective feedback in these cases, and we approach this goal through Reinforcement Learning (RL).

We implement an RL based approach for training LLMs to improve their corrective skills. To do so, our training data consists of several controlled generation tasks. During training we let the model interact with corrective feedback over multiple iterations, and provide a reward corresponding to the provided feedback. Specifically, the delivered reward corresponds to the best feedback obtained in the interaction session. To understand why this is the reward chosen, consider a controlled generation task, where the corrective feedback says how many constraints are satisfied. Then, one can always choose the output that satisfied the most constraints. See Figure 1 for an illustration of an interaction session and corresponding reward.

For the above reward scheme, we train the model using RL, to obtain a model that can better use corrective signals. Our first observation is that on the tasks that the model was trained on, there is a significant improvement in performance, as compared to baselines that are not trained to use feedback. An even more surprising finding is that these trained models also improve when using feedback in tasks they were not trained on. This supports our conjecture that LLMs can improve on the "meta-skill" of using corrective feedback.

Our results demonstrate the significant potential of teaching LLMs to effectively incorporate feedback into their output generation processes. By demonstrating improvements in controlled generation tasks, we highlight how LLMs can learn to refine their outputs based on corrective input, leading to more accurate and reliable results. This suggests that the ability to utilize feedback is not merely a task-specific enhancement but may represent a more generalizable skill that would demonstrate gains across a wide range of tasks.

Figure 1: An example of the CORGI setup. We consider a dialogue between a generator and a critic. Here the generator is tasked with completing a given sentence in precisely four words, with the final word being "first". The critic evaluates the responses of the generator, providing both feedback and a score (illustrated as a star rating). The LLM receives a reward based on the highest score assigned by the critic throughout the dialogue history. To prioritize improvement on the more challenging length constraint in this case, we set the constraint weights to 80% for length and 20% for the last word constraint.

CORGI Setup and Model

This section outlines our proposed approach: Controlled Generation with RL for Guided Interaction, or CORGI.

Problem Definition

In controlled generation tasks (See et al. 2019; Gehman et al. 2020; Sun et al. 2023) the model is presented with specific constraints or requirements that define the desired characteristics of the generated text. For these tasks we assume access to an automated critique environment. Our primary aim is to teach an LLM to effectively incorporate critique feedback, while bypassing the need for labeled data. We aim to train a model capable of generating text that adheres to these constraints, even in scenarios where the model has not been directly trained on the task and its critique system.

Formally, consider a set of M controlled-generation tasks $\{\mathcal{T}_i\}_{i=1}^{M}$. For each task $\mathcal{T}_j$, let $D_{\mathcal{T}_j} = \{x_i\}_{i=1}^{N_j}$ denote an unlabeled dataset of controlled generation prompts,[1] where $x \in \mathcal{X}$ is the task input. Additionally, for each task, let $C_{\mathcal{T}_j}$ denote the automated critique. This critique is defined as a function $C_{\mathcal{T}} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R} \times \mathcal{Z}_{\mathcal{T}}$ which takes as an input the task's input and a generated output $y \in \mathcal{Y}$, and returns a score $r \in \mathbb{R}$ and a textual feedback $z \in \mathcal{Z}_{\mathcal{T}}$. For example, in Figure 1 the critic offers textual feedback, notifying the generator that it used the wrong last word or that the output was too long. This feedback is accompanied by a score, illustrated as a star rating in the figure.

Our objective is to train a model on the M available task datasets using their critique environment. Importantly, we would like for the model to also perform well on new tasks $\mathcal{T}'$ that have their own critic feedback $C_{\mathcal{T}'}$ and are not used for training.

In what follows, we first describe the iterative use of critiques, and the corresponding reward the model receives, and then describe how this reward can be optimized using RL.

[1]Unlabeled here means no correct output is provided for x_i.

Iterative Use of Critiques

We consider the setting where the model dynamically receives feedback (Shinn et al. 2024; Yao et al. 2022; Madaan et al. 2023). In this framework, the model receives real-time textual feedback from the critic, allowing it to adjust and refine its output. The key aspect of our setting is that the critic is invoked every time the model produces a new response, so that the model gets multiple attempts to produce a good answer. We limit the number of interactions to K. After K iterations (or sooner, if the model successfully completes the task), the model returns the response that received the highest score from the critic.

Reward Definition Our key goal is to train LLMs to improve their performance when using critiques in an iterative fashion. We use an RL approach towards this end. The key component in such a training approach is the reward that the model receives in each session. In our setting it is clear that this reward is simply the best critique score obtained in the session, since this is the quality of the returned output.[2] Formally, in each session, the generator produces up to K outputs, with each output o_i being evaluated by the critic and assigned a score s_i. The session reward is then defined as:

$$r = \max_{i=0,\ldots,K} s_i \tag{1}$$

Using RL to Improve the Use of Feedback

To maximize the expected reward across sessions, we use reinforcement learning, specifically Proximal Policy Optimization (PPO) (Schulman et al. 2017), a method proven effective in Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al. 2022). Building on previous applications of reinforcement learning in large language models (Ouyang et al. 2022; Achiam et al. 2023), we define the actions $a_i \in \mathcal{A}$ as the tokens selected from the vocabulary $\mathcal{V}$

[2]The best critique score may not come from the last output. For example in a length constrained generation task where the goal is to product 6 words, if the model produces outputs with lengths 2, 5, and 3, the second output would be selected.

by the policy, while the state corresponds to the prompt presented to the model at each time step.

We further integrate reinforcement learning into the CORGI framework, enhancing its ability to adapt and optimize based on real-time feedback. We designed a generalized environment simulating a conversation with automated critique for reinforcement learning purposes. The trajectory is defined by the interactions generated by the policy with the automatic critique. Formally, a chat between the LLM and the critic is represented as $(x, a_{0_0}, ..., a_{0_{n_1}}, z_1, a_{1_1}, ..., z_t, a_{t_0}, ..., a_{t_{n_t}})$ where x is the task specific prompt, $\{z_i\}_{i=1}^{t}$ is the textual feedback from a critic $\mathcal{C}$. The actions $\{a_{i_j}\}_{j=1}^{n_i}$ correspond to the tokens generated by the LLM in the i^{th} attempt to generate the output defined in the prompt.[3] An example for a chat can be seen in Figure 1. Episodes start with a prompt x sampled from the dataset and end when the output achieves a perfect score or after K attempts, with each output constrained to T tokens or terminated by an EOS token.

Our goal is then to maximize the PPO objective function:

$$J(\theta) = \mathbb{E}[\min(r(\theta)\hat{A}_{\theta_{\text{ref}}}(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{\text{ref}}}(s,a))] \,, \tag{2}$$

where $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{ref}}}(a|s)}$ is defined as the probability ratio between the new policy and the reference policy, $\hat{A}_{\theta_{\text{ref}}}(s,a) = \hat{Q}(s,a) - \hat{V}(s)$ is the estimated advantage function using a value head, and ϵ is a hyperparameter. To distinguish between action tokens (outputs) and non-action tokens (prompt and feedback), we mask non-action tokens by setting their reward and value functions to zero.

In this way, the CORGI framework utilizes both textual feedback and the critic's scoring, teaching the policy to adjust to the textual feedback in real-time and enhancing the reward signal by incorporating the feedback text into the state.

Experiments

In what follows we apply our CORGI approach to several controlled generation problems.

Data

We conduct experiments across eleven diverse tasks. We include five tasks inspired by those introduced by Sun et al. (2023): Sentiment Reviews Generation, Story Generation, Rationale Generation, Numerical Planning, and Controlled Paraphrase Generation. Due to the unavailability of the original data and code, we reconstructed the dataset.

We further evaluated our approach on a broader set of tasks, including Style Transfer (Pryzant et al. 2020), CommonGen-lite (Lin et al. 2020), Program Synthesis (Numeric category) (bench authors 2023), MBPP (sanitized subset) (Austin et al. 2021), and CommonGen-Hard (Madaan et al. 2023). Additionally, we introduced two new tasks: Panagram and Clustering, described below.

[3]For example, if the prompt is "Generate a sentence with 6 words" then a_3. would be the third try of generating this sentence, after two previous tries that received feedback.

In the Panagram task, the model is required to generate a word using a given list of letters. The generated word must include all specified letters, with each letter used at least once and possibly multiple times.

The Clustering task explores a more algorithmic challenge. Here, the input is a list of student names, and the model must group the students into clusters of two or more, taking into account their preferences, namely which students they prefer or do not want to be grouped with. The critic provides as feedback the constraints that were violated.

Further details on data construction and the reward system for all tasks are provided in the arxiv version of this paper (Bezalel, Orgad, and Globerson 2024).

For simplicity, we normalize all rewards to be in the $[0, 1]$ range, where 1 indicates a perfect response. For all these tasks, we include a textual response by the critic, which is provided to the generator whenever the reward is less than the maximal value of one.

We categorize the tasks into two groups:

1. *Source Tasks* - These tasks are used for training with RL. We used the following source tasks: Sentiment Reviews Generation, Story Generation, Rationale Generation, Numerical Planning, and Controlled Paraphrase Generation.
2. *Target Tasks* - These tasks are only used for evaluation and are not used during RL. Namely, we use these to evaluate how well the model transfers to problems it wasn't trained on. We used the following target tasks: Style Transfer, Clustering, Panagram, and CommonGen-lite. Additionally, we introduced a subgroup of Llama-3-specific target tasks, which includes Program Synthesis, MBPP, and CommonGen-Hard. These tasks were specifically evaluated with Llama-3 due to poor performance by Llama-2.

Critique Construction

The critique function evaluates how well a candidate output y aligns with a task description x, producing a numerical score (0–1) and textual feedback. The design of the critique varies by task and typically involves rule-based programs or classifiers using template-based feedback. For instance, in Sentiment Reviews Generation, the critic simply employs a classifier that checks if the review matches the product name and star rating. Full details on all critiques are provided in Bezalel, Orgad, and Globerson (2024).

Pre-processing

For each task, we created a prompt outlining the constraints and including two few-shot examples. Additionally, we generated 7,500 training prompts and 500 validation prompts per task. For Rationale Generation, Controlled Paraphrase Generation, Common-Gen lite, Program Synthesis (numeric category), MBPP (sanitized subset), Common-Gen Hard tasks we used the predefined train/dev/test splits - sampling the training prompts and validation prompts from their corresponding splits. For tasks without predefined train/dev/test splits, we additionally produced 1,000 test prompts.

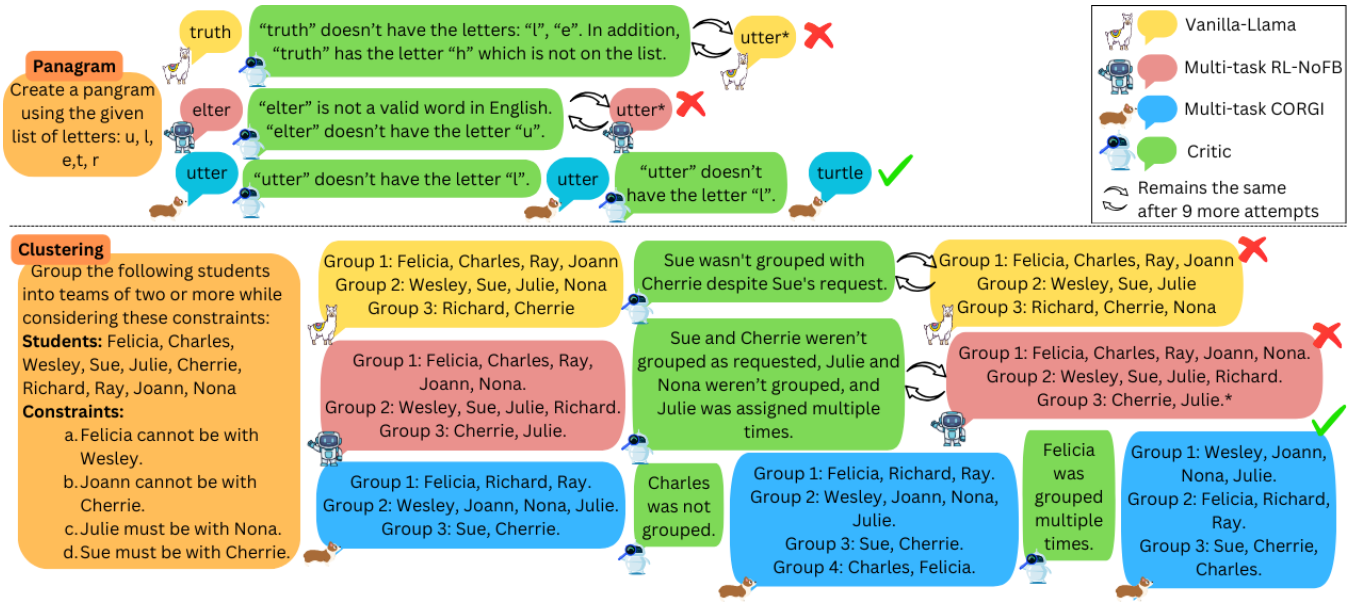

Figure 2: Meta-learning examples showing responses of Vanilla Llama, RL-NoFB, and CORGI on the Panagram and Clustering tasks, with the latter two models trained jointly on multiple source tasks. CORGI arrives at a correct output after several iterations, whereas the other models get stuck repeating a suboptimal solution.

Experimental Setup

Baselines Our study utilizes the models Llama-2-7b-chat (Touvron et al. 2023) and Llama-3-8b-instruct (AI@Meta 2024) as the initial checkpoints to which RL is applied.[4] We examine two distinct baseline configurations:

1. **Vanilla-Llama** - The Llama models, without any additional training. This setup aligns with related work (Shinn et al. 2024; Madaan et al. 2023) where feedback is derived from an external source.
2. **RL-NoFB** - This model serves as a reinforcement learning baseline without receiving feedback during training. The model is given a controlled generation prompt and generates a response. A critique is applied to this response, and its value is used as a reward for the PPO algorithm. Unlike CORGI, this model does not receive textual feedback during training nor does it make multiple generation attempts.

During evaluation, we allow the baseline models to make multiple attempts and provide them with the same feedback as CORGI to assess their ability to utilize this feedback. For each model, we select the attempt with the highest score according to the critique. We note that such multiple attempts can only make the baseline stronger.

Training We trained CORGI and RL-NoFB using Llama-2-7b-chat and Llama-3-8b-instruct, with Proximal Policy Optimization (PPO). The CORGI framework was implemented using the TRL (von Werra et al. 2020) library, which we also utilized for RL-NoFB by setting the number of attempts to one. The Adam optimizer was employed with a learning rate of 10^{-5} and Adaptive KL control (Ziegler et al. 2019), with initial coefficients set to 0.05 for Llama-2 and 0.075 for Llama-3. Due to training instability, the KL coefficient was adjusted to 0.3 for the rationale generation task in Llama-3. For CORGI training, the number of attempts was limited to four per prompt due to computational constraints. Training was conducted on a single NVIDIA A100-SXM4-80GB GPU, taking 4 days for the multi-task setting and 12-24 hours for the single-task setting.

As mentioned above, we were interested in the ability of the model to transfer to tasks that were not seen during training, and the advantage of training on multiple tasks. Thus, we considered two different training scenarios:

1. **Single-Task Training:** RL-NoFB and CORGI are trained on individual tasks, producing five models.
2. **Multi-Task Training:** RL-NoFB and CORGI are jointly trained on data from all tasks, creating one model.

Evaluation

We evaluate the aforementioned settings on the eleven tasks. When evaluating on the source tasks, in the single-task setting, we evaluate each model on the task it was trained on. To assess the meta-learning capabilities of the multi-task models, we include evaluations on the Target Tasks.

During inference, we employ the same automated critique environment used during CORGI training. Each generator, including the baseline models, uses greedy decoding and receives 10 attempts for Llama-2 and 20 attempts for Llama-3,[5] incorporating the critic's feedback to complete the tasks.

[4] Our focus on RL excludes models that cannot be trained.

[5] Llama-3 has more attempts due to its larger context size of 8192 tokens, compared to Llama-2's 4092 tokens.

Model	Training	Setting	Source Tasks	Target Tasks	Overall
Llama 2	None	Few-shot	68.8	55.4	62.8
	Single-Task	RL-NoFB	71.4	53.9	63.7
		CORGI	**73.7**	**54.5**	**65.2**
	Multi-Task	RL-NoFB	70.7	52.4	62.5
		CORGI	**74.6**	**58.5**	**67.5**
Llama 3	None	Few-shot	86.3	84.2	84.3
	Signle-Task	RL-NoFB	87.6	83.1	85.6
		CORGI	**92.8**	**84**	**88.9**
	Multi-Task	RL-NoFB	87.3	84	85.9
		CORGI	**90.8**	**85.2**	**88.3**

Table 1: Results on controlled generation tasks. The table shows performance on source tasks and target tasks, excluding Llama-3-specific target tasks (results for these tasks reported in Figure 3). All results are for the best generated response after 10 attempts for Llama-2 and 20 attempts for Llama-3. Standard deviation is estimated with bootstrap. The first row for each model corresponds to the Vanilla-Llama model. The second row and third rows (Single-Task) show the average results for five different models trained on a single training task. The fourth and fifth rows (Mutli-Task) show the results of a single model per training setting, each trained on five source tasks simultaneously. We provide the average success rate for the source tasks (column 3), the average success rate for the target tasks (column 4), and the overall average success rate for all nine tasks (column 5). The results per task are reported in Bezalel, Orgad, and Globerson (2024)

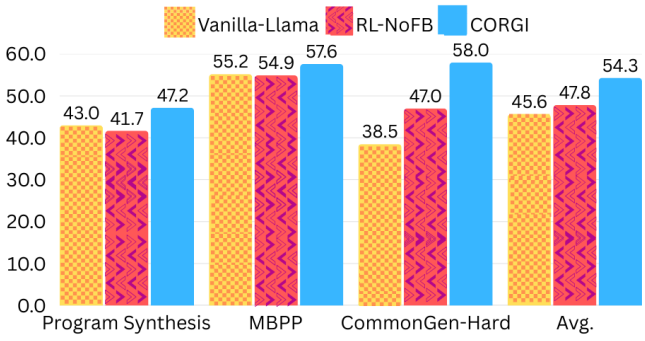

Figure 3: Multi-task performance on the Llama-3-Specific Target Tasks shows CORGI outperforms the baselines, benefiting from transfer learning.

Results and Analysis

We report the main results in Table 1 and Figure 3. For all tasks we report the success rate metric, where success is defined as satisfying all the constraints of the task. Our experiments show that CORGI significantly improves the performance over the baselines, in both in-task and meta-learning experiments.

Source Tasks Results using Single Task Training

In Table 1 we present the source task results for the single-task trained models. For both Llama models, we observe that both RL-NoFB and CORGI outperform the Vanilla-Llama settings. Notably, CORGI training proves to be significantly more effective than RL-NoFB, showing an average improvement of 2% in Llama-2 and 5% in Llama-3. These results indicate that CORGI training effectively leverages critique feedback to substantially enhance generator performance.

Figure **??** shows results as a function of the number of critique feedback iterations. It can be seen that access to multiple critique feedbacks is beneficial for all three models, with performance further increasing after RL training. Even though the RL-NoFB model was not trained with feedback, it still benefits from these, though it did not perform as well as CORGI, which was specifically trained with feedback. We also observe that the performance gap between RL-NoFB and CORGI increases with the number of attempts, particularly in the initial attempts, indicating that CORGI leverages feedback more effectively.

Multi Task and Meta Learning Results

Source Tasks Results The results for multi-task training on source tasks are presented in Table 1. Both the RL-NoFB and CORGI models show improvement compared to the Vanilla-Llama setting, with CORGI consistently outperforming RL-NoFB.

For Llama-2, multi-task training yields results comparable to single-task training, with a slight improvement in the CORGI setting. Conversely, in Llama-3, single-task training appears to be more effective than multi-task training in the CORGI setting. The limited improvement of multi-task training on the source tasks is expected, as it may be more beneficial for the model to focus specifically on the tasks it is being evaluated on.

Figure **??** shows performance over multiple feedback iterations. Initially, RL-NoFB performs comparably to (Llama-3) or outperforms (Llama-2) CORGI due to its more greedy approach. However, with additional attempts, CORGI improves more rapidly, increasing the performance gap between the two methods. This behavior is also evident in the Llama-3 setting. These results indicate that CORGI benefits more from iterative feedback.

Target Tasks Results We next discuss the results when applying models to tasks they were not trained on. Here one might expect that training on multiple tasks would improve performance, which we indeed observe.

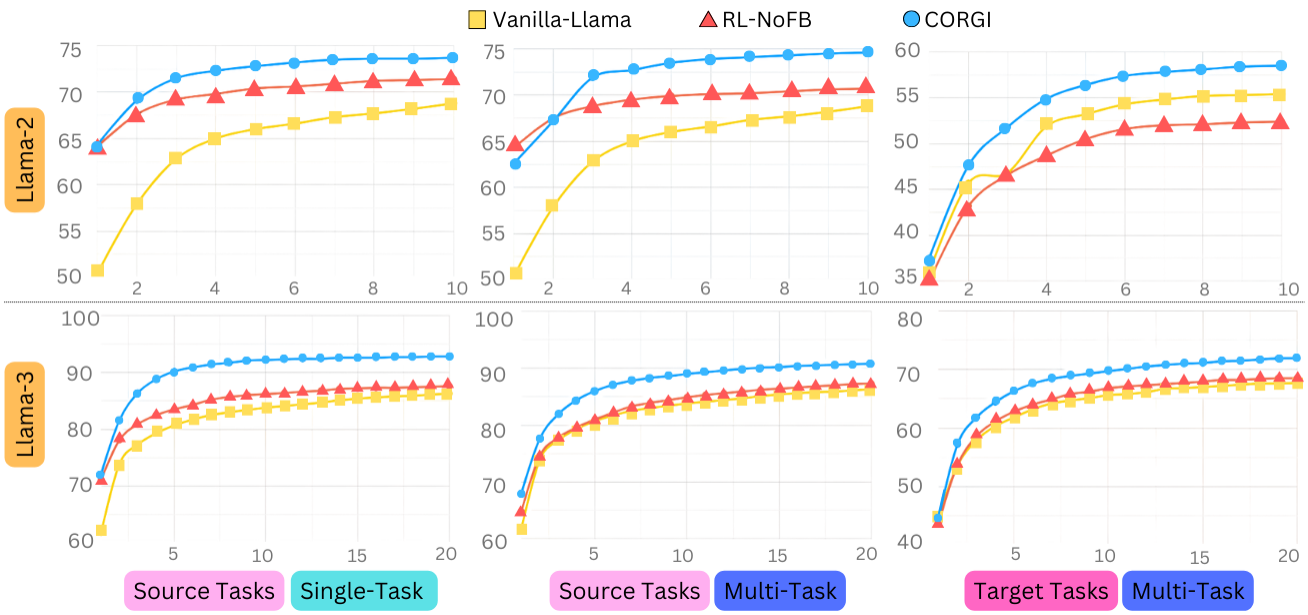

Figure 4: Success rates for Llama-2 (10 attempts) and Llama-3 (20 attempts) models in single-task and multi-task training. CORGI consistently achieves the best performance in later attempts. Additionally, the performance gap between CORGI and the baselines widens over successive attempts, highlighting CORGI's superior learning and adaptation capabilities.

Table 1 show that multi-task CORGI models outperform the single-task ones, suggesting that multi-task training improves meta-learning. For example, multi-task CORGI improves over single-task by four percent (from 54.5 to 58.5) for Llama-2, and by 1.2% for Llama-3. For the RL-NoFB model, there is no improvement for multi-task in Llama-2 and 0.9% improvement for Llama-3. As illustrated in Figure **??**, Multi-Task CORGI effectively adapts new tasks not seen during training. Furthermore, the performance gap increases with the number of attempts.

Ablation Study

We conducted an ablation study to evaluate the significance of feedback information and the design choices in our framework. All experiments were conducted using variations of CORGI training on the Llama-2-7B-chat model, applied to the numerical planning task.

Detailed Feedback We tested what happens when one replaces the feedback with binary information, removing any useful details (such as the number of words in the generated output or its last word). Specifically, for imperfect outputs we use the feedback "Your output is incorrect. Please try again.". We use this feedback in both train and evaluation. In Figure 5 it can be seen that the model with binary feedback performs considerably worse than the one with more detailed feedback. Additionally, the CORGI model still improves considerably over the Vanilla Llama baseline with binary feedback, indicating that the former has learned to use the binary feedback.

Reward Design We examined the impact of reward design by comparing the performance of rewarding the last generation attempt to our original approach of rewarding the attempt with the highest score, while still selecting the best output across attempts at inference. As can be seen in Figure 5, rewarding the last attempt led to some improvement over the baseline, but it underperformed compared to using the maximum reward approach.

Multi-Turn training We evaluated the importance of multi-turn training by comparing it to single-turn training, where only one feedback instance is provided. We observed a significant drop in performance, suggesting that training multiple turns is beneficial. This is intuitively plausible, as it better aligns with inference-time, where multiple improvement steps can be done.

Related Work

Controlled generation There are various approaches that tackle controlled generation. Some approaches (Anderson et al. 2017; Hokamp and Liu 2017; Post and Vilar 2018; Lu et al. 2021, 2022) proposed constrained search algorithms that limit the lexical constraints by altering the search space. These approaches are limited to lexical constraints. Another more flexible approach is score-based sampling, where constraints are represented via a differentiable score function. For instance, Liu et al. (2022) represented lexical constraints via a differentiable n-gram matching function.

A more closely related approach to our method is Han et al. (2023), where a verification module is trained to provide feedback to a generation model, aiding in the refinement of graph-based tasks. In contrast, our approach involves training the generator LLM while incorporating feedback, whereas the generator LLM in Han et al. (2023) re-

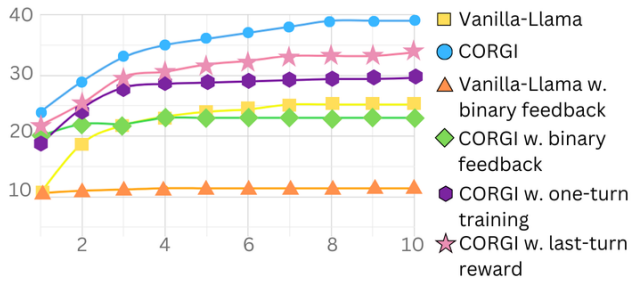

Figure 5: Ablation study results for the numerical planning task: CORGI (blue) outperforms all configurations, showcasing the benefits of detailed feedback, multi-turn training, and maximum reward strategy.

mains frozen without additional training. Zhou et al. (2023) showcases the effectiveness of meta-learning in controlled generation. This approach fine-tunes a model on multiple controlled generation tasks, allowing it to adapt to new constraints without retraining. In our work, we take this concept further by creating an interactive environment where the model receives real-time feedback from a critique system, allowing iterative refinements, leading to improved generalization when encountering new constraints.

Reasoning with Feedback Loops Feedback loops have recently been used to allow models to refine their outputs. Akyürek et al. (2023) trains a critique mechanism to automate feedback provided to the generator model, aiding in its refinement process. Paul et al. (2023) fine-tunes both critique and generator models. In this method, the generator is trained to produce intermediate reasoning steps, which are then evaluated by the critic model, facilitating a continuous improvement process. This method required annotated data for the intermediate steps for training both the critique and generator models. Besta et al. (2024) and Long (2023) introduce the GoT and ToT frameworks, which integrate feedback mechanisms into the generation process, enhancing the COT process. Miao, Teh, and Rainforth (2023) allows LLMs to examine and evaluate their reasoning steps. Chen and Chang (2023) uses different language models as auxiliary roles, such as checkers and sorters, to help the main language model avoid erroneous and inefficient actions. Yao et al. (2022) proposes constructing prompts using thought-act-observation triplets aiming to enhance reasoning, planning, and interaction with the environment. Shinn et al. (2024) uses a dynamic feedback system, capturing recent feedback and persistent long-term storage to retain key insights, providing a robust mechanism for ongoing refinement. Inspired by these approaches, our work aims to enhance feedback utilization in the LLM refinement process through reinforcement learning. In addition, our approach does not require labeled data and allows for multiple generations during training.

Alignment with reinforcement learning Reinforcement learning techniques play a critical role in optimizing models through feedback-driven reward signals. Ouyang et al. (2022) and Lee et al. (2023) reinforce a supervised fine-tuned (SFT) model. Specifically, they used a PPO approach for aligning the model with human feedback by training a reward module based on human preferences. Lee et al. (2023) replace human preference labels with those of the model itself - the model is prompted to evaluate its own predictions post-supervision, in consideration of human values and preferences. These approaches primarily focus on refining models through explicit reward signals. Our method integrates both textual critique and reinforcement learning by incorporating a conversational critique mechanism which complements the reinforcement-based rewards.

Conclusion

We present an approach for improving the capability of LLMs to use feedback in an interactive manner. Our key observation is that a model can learn to use such interactive feedback via reinforcement learning. A key advantage of our scheme is that it does not require labeled data (i.e., examples of correct generation). Instead it just employs a critic which can score generations.

Our results show that models indeed benefit from this training scheme. Not only can they improve on tasks that were used during RL, but they also improve on very different tasks (e.g., clustering and Panagram) that they were not trained on. This suggests that CORGI training indeed endowed the model with a capability to receive a textual critique in an interactive fashion, and continuously improve "on tape" using this critique.

Our work raises several questions for followup. One is the mechanism via which the model learns to use the reward. Namely, how does the textual feedback translate into the action of improving generation. It will be interesting to study this using tools recently employed in understanding in-context learning (e.g., Hendel, Geva, and Globerson 2023; Todd et al. 2024). A related question is optimizing over the textual feedback. Namely, is it possible to effectively optimize over the specific text that is used in feedback (e.g., similar to soft-prompt tuning Lester, Al-Rfou, and Constant (2021)).

Our approach can be used to significantly enhance the quality of controlled generation of text, thus improving the applicability of LLMs in domains such as assistants and education.

Selecting and designing critiques is also a key consideration for real-world applications, where feedback may vary across domains or user contexts. While our results demonstrate that CORGI performs well across tasks with varying feedback, for general scenarios where implementing a custom critique is impractical, an LLM can be prompted to function as the critic. However, this approach may be more computationally expensive.

Further research could explore using human input in addition to the critique input to better adjust the LLM to user preferences. It will also be interesting to investigate more algorithmic tasks where critic not only provide scores but also suggest strategies for improvement. Exploring these directions may yield a richer understanding of how models can learn and adapt in more complex, real-world scenarios.

Acknowledgments

This project received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant ERC HOLI 819080). It was also supported by the Tel Aviv University Center for AI and Data Science (TAD).

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI@Meta. 2024. Llama 3 Model Card. Accessed: 2024-06-13.

Akyürek, A. F.; Akyürek, E.; Madaan, A.; Kalyan, A.; Clark, P.; Wijaya, D.; and Tandon, N. 2023. Rl4f: Generating natural language feedback with reinforcement learning for repairing model outputs. *arXiv preprint arXiv:2305.08844*.

Anderson, P.; Fernando, B.; Johnson, M.; and Gould, S. 2017. Guided Open Vocabulary Image Captioning with Constrained Beam Search. In Palmer, M.; Hwa, R.; and Riedel, S., eds., *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 936–945. Copenhagen, Denmark: Association for Computational Linguistics.

Austin, J.; Odena, A.; Nye, M.; Bosma, M.; Michalewski, H.; Dohan, D.; Jiang, E.; Cai, C.; Terry, M.; Le, Q.; et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

bench authors, B. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Podstawski, M.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Niewiadomski, H.; Nyczyk, P.; et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17682–17690.

Bezalel, L.; Orgad, E.; and Globerson, A. 2024. Teaching Models to Improve on Tape. arXiv:2411.01483.

Chen, P.-L.; and Chang, C.-S. 2023. Interact: Exploring the potentials of chatgpt as a cooperative agent. *arXiv preprint arXiv:2308.01552*.

Gehman, S.; Gururangan, S.; Sap, M.; Choi, Y.; and Smith, N. A. 2020. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. In Cohn, T.; He, Y.; and Liu, Y., eds., *Findings of the Association for Computational Linguistics: EMNLP 2020*, 3356–3369. Online: Association for Computational Linguistics.

Han, J.; Collier, N.; Buntine, W.; and Shareghi, E. 2023. Pive: Prompting with iterative verification improving graph-based generative capability of llms. *arXiv preprint arXiv:2305.12392*.

Hendel, R.; Geva, M.; and Globerson, A. 2023. In-Context Learning Creates Task Vectors. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Findings of the Association for Computational Linguistics: EMNLP 2023*, 9318–9333. Singapore: Association for Computational Linguistics.

Hokamp, C.; and Liu, Q. 2017. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In Barzilay, R.; and Kan, M.-Y., eds., *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1535–1546. Vancouver, Canada: Association for Computational Linguistics.

Lee, H.; Phatale, S.; Mansoor, H.; Lu, K.; Mesnard, T.; Bishop, C.; Carbune, V.; and Rastogi, A. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Lin, B. Y.; Zhou, W.; Shen, M.; Zhou, P.; Bhagavatula, C.; Choi, Y.; and Ren, X. 2020. CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning. In Cohn, T.; He, Y.; and Liu, Y., eds., *Findings of the Association for Computational Linguistics: EMNLP 2020*, 1823–1840. Online: Association for Computational Linguistics.

Liu, G.; Yang, Z.; Tao, T.; Liang, X.; Bao, J.; Li, Z.; He, X.; Cui, S.; and Hu, Z. 2022. Don't Take It Literally: An Edit-Invariant Sequence Loss for Text Generation. In Carpuat, M.; de Marneffe, M.-C.; and Meza Ruiz, I. V., eds., *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2055–2078. Seattle, United States: Association for Computational Linguistics.

Long, J. 2023. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*.

Lu, P.; Peng, B.; Cheng, H.; Galley, M.; Chang, K.-W.; Wu, Y. N.; Zhu, S.-C.; and Gao, J. 2023. Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 43447–43478. Curran Associates, Inc.

Lu, X.; Welleck, S.; West, P.; Jiang, L.; Kasai, J.; Khashabi, D.; Le Bras, R.; Qin, L.; Yu, Y.; Zellers, R.; Smith, N. A.; and Choi, Y. 2022. NeuroLogic A*esque Decoding: Constrained Text Generation with Lookahead Heuristics. In Carpuat, M.; de Marneffe, M.-C.; and Meza Ruiz, I. V., eds., *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 780–799. Seattle, United States: Association for Computational Linguistics.

Lu, X.; West, P.; Zellers, R.; Le Bras, R.; Bhagavatula, C.; and Choi, Y. 2021. NeuroLogic Decoding: (Un)supervised Neural Text Generation with Predicate Logic Constraints. In Toutanova, K.; Rumshisky, A.; Zettlemoyer, L.; Hakkani-Tur, D.; Beltagy, I.; Bethard, S.; Cotterell, R.; Chakraborty,

T.; and Zhou, Y., eds., *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4288–4299. Online: Association for Computational Linguistics.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; Gupta, S.; Majumder, B. P.; Hermann, K.; Welleck, S.; Yazdanbakhsh, A.; and Clark, P. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 46534–46594. Curran Associates, Inc.

Miao, N.; Teh, Y. W.; and Rainforth, T. 2023. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *arXiv preprint arXiv:2308.00436*.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

Paul, D.; Ismayilzada, M.; Peyrard, M.; Borges, B.; Bosselut, A.; West, R.; and Faltings, B. 2023. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*.

Post, M.; and Vilar, D. 2018. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In Walker, M.; Ji, H.; and Stent, A., eds., *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1314–1324. New Orleans, Louisiana: Association for Computational Linguistics.

Pryzant, R.; Martinez, R. D.; Dass, N.; Kurohashi, S.; Jurafsky, D.; and Yang, D. 2020. Automatically neutralizing subjective bias in text. In *Proceedings of the aaai conference on artificial intelligence*, volume 34, 480–489.

Schick, T.; Dwivedi-Yu, J.; Dessi, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 68539–68551. Curran Associates, Inc.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

See, A.; Roller, S.; Kiela, D.; and Weston, J. 2019. What makes a good conversation? How controllable attributes affect human judgments. In Burstein, J.; Doran, C.; and Solorio, T., eds., *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)*, 1702–1723. Minneapolis, Minnesota: Association for Computational Linguistics.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Sun, J.; Tian, Y.; Zhou, W.; Xu, N.; Hu, Q.; Gupta, R.; Wieting, J.; Peng, N.; and Ma, X. 2023. Evaluating Large Language Models on Controlled Generation Tasks. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 3155–3168. Singapore: Association for Computational Linguistics.

Todd, E.; Li, M. L.; Sharma, A. S.; Mueller, A.; Wallace, B. C.; and Bau, D. 2024. In *Function Vectors in Large Language Models*, volume abs/2310.15213.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

von Werra, L.; Belkada, Y.; Tunstall, L.; Beeching, E.; Thrush, T.; Lambert, N.; and Huang, S. 2020. TRL: Transformer Reinforcement Learning. https://github.com/huggingface/trl. Accessed: 2024-02-18.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Zhou, W.; Jiang, Y. E.; Wilcox, E.; Cotterell, R.; and Sachan, M. 2023. Controlled Text Generation with Natural Language Instructions. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 42602–42613. PMLR.

Ziegler, D. M.; Stiennon, N.; Wu, J.; Brown, T. B.; Radford, A.; Amodei, D.; Christiano, P.; and Irving, G. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.