

RL4F: Generating Natural Language Feedback with Reinforcement Learning for Repairing Model Outputs

Afra Feyza Akyürek¹

Ekin Akyürek²

Ashwin Kalyan⁴

Peter Clark⁴

Derry Wijaya^{1,3}

Niket Tandon⁴

¹Boston University ²MIT CSAIL ³Monash University Indonesia

⁴Allen Institute for Artificial Intelligence

Abstract

Despite their unprecedented success, even the largest language models make mistakes. Similar to how humans learn and improve using feedback, previous work proposed providing language models with *natural language feedback* to guide them in repairing their outputs. Because human-generated critiques are expensive to obtain, researchers have devised learned critique generators in lieu of human critics while assuming one can train downstream models to utilize generated feedback. However, this approach does not apply to black-box or limited access models such as ChatGPT, as they cannot be fine-tuned. Moreover, in the era of large general-purpose language agents, fine-tuning is neither computationally nor spatially efficient as it results in multiple copies of the network. In this work, we introduce RL4F (Reinforcement Learning *for* Feedback), a multi-agent collaborative framework where the critique generator is trained to maximize end-task performance of GPT-3, a fixed model more than 200 times its size. RL4F produces critiques that help GPT-3 revise its outputs. We study three datasets for action planning, summarization and alphabetization and show relative improvements up to 10% in multiple text similarity metrics over other learned, retrieval-augmented or prompting-based critique generators.¹

1 Introduction

Correcting model outputs is a pressing challenge in natural language generation (Ribeiro et al., 2018; Reid and Neubig, 2022), emerging across many use-cases such as style transfer (Mallinson et al.,

¹A significant portion of this work was done while Feyza was an intern at Allen AI. Correspondences to Afra Feyza Akyürek (akyurek@bu.edu).

Action Planning

input (x):

put soap in dishwasher

model output (\hat{y}):

1. open dishwasher door
2. grab some dish soap
3. put soap in dispenser

critique (c):

One cannot put soap in the dispenser without first opening it.

corrected output (\hat{y}_{new}):

1. open dishwasher door
2. grab some dish soap
3. open dispenser
4. put soap in dispenser

Topic-Based Summarization

input (x):

Passage: When did it all begin? Where it always does I suppose, with the vows [...]
Question: Summarize the plot of the story.

model output (\hat{y}):

Michael sits in the waiting room, thinking about his family [...] he was betrayed by his father Bill [...]

critique (c):

Bill wasn't Michael's father.

corrected output (\hat{y}_{new}):

Michael sits in the waiting room, thinking about his marriage [...] he was betrayed by his friend Bill [...]

Figure 1: Two examples for action planning (Tandon et al., 2021) and summarization (Saunders et al., 2022) tasks show-case a scenario where initial predictions by a learned model (\hat{y}) are incorrect. Human-written critiques (c) indicate errors in model outputs. While humans can reliably critique each other, machines lack such ability. This paper studies a multi-agent collaborative framework where one language model can generate critiques to improve its peer's performance.

2020; Malmi et al., 2022), grammatical (Lichtarge et al., 2019) or factual error correction (Mitchell et al., 2022b), debiasing and detoxification (Schick et al., 2021). Unlike humans who can understand natural language feedback and improve using the information, most of the previous work relied on sequence tagging (Reid and Neubig, 2022), retraining from scratch (Sun et al., 2019) or parameter editing (Mitchell et al., 2022a) to repair model predictions.

Recently, researchers show that large language models can correct their answer given more sophisticated feedback formulated in natural language (Schick et al., 2022; Saunders et al., 2022). For example, in Fig. 1, we present sample feedback for two tasks. Both of these examples exemplify the case where the initial model outputs \hat{y} have flaws. In topic-based summarization, an automatically generated summary of a story involves factually incorrect statements such as “... *he was betrayed by his father Bill ...*” where an appropriate critique is “*Bill is not Michael’s father*”. In action planning, given a goal x , the objective is to generate a set of steps y to achieve the goal. The initial sequence of actions in Fig. 1, denoted by \hat{y} , has a missing a step. The human-written natural language critiques c describe the ways in which \hat{y} ’s are incorrect and \hat{y}_{new} denotes the corrected prediction conditioned on the critique. Note that in many situations helpful critiques do not necessarily reproduce an entire answer—they may simply point out one way in which the answer could be improved.

Researchers use crowd-sourcing to collect critiques for model outputs (Saunders et al., 2022). However, collecting feedback from humans is infeasible in an online setting where a model is required to produce a rapid stream of outputs. The goal of this paper is to shed light on whether the task of critiquing language model predictions can be effectively passed on to an external agent while keeping the language model itself intact.

Our multi-agent collaborative framework involves two language models where one model’s job is to criticize the other as the latter performs a task of interest, such as summarization. This setting comprises a task model, denoted by LM_{task} , which learns the mapping from an input x (e.g. passage) to a ground truth output y (e.g. summary); and a critiquing model $\text{LM}_{\text{critique}}$ which provides natural language critiques for LM_{task} ’s outputs $\hat{y} \sim \text{LM}_{\text{task}}(x)$. The framework can additionally involve a separate model (say $\text{LM}_{\text{refine}}$) for repairing model outputs conditioned on critiques. We follow past work (Schick et al., 2022), and merge LM_{task} and $\text{LM}_{\text{refine}}$ into a single model. Hence, in addition to predicting y given x , LM_{task} is also tasked to improve its initial output conditioned on a critique \hat{c} sampled from $\text{LM}_{\text{critique}}(x, \hat{y})$.

We introduce RL4F (Reinforcement Learning for Feedback Generation), a cascade (Dohan et al., 2022) of two language models for automatic cri-

tique generation and refinement. RL4F is trained to maximize target task performance of LM_{task} by learning to provide critiques for its outputs via $\text{LM}_{\text{critique}}$. RL4F advances retrieval-based methods with learned critique generation (Madaan et al., 2022). Unlike previous work which teaches LM_{task} to read a crowd-sourced set of critiques (Schick et al., 2022; Saunders et al., 2022), RL4F learns the particular set of critiques that will steer LM_{task} into improving its predictions without requiring any updates to LM_{task} parameters. Treating LM_{task} as fixed is especially important in era of limited-access large language models which are costly, if not impossible, to fine-tune.

RL4F is illustrated in Fig. 2(a,c). Previous work demonstrate that language models smaller than roughly 50 billion parameters lack the ability to understand and act upon a natural language critique (Saunders et al., 2022; Bai et al., 2022). Therefore, we chose GPT-3 as the LM_{task} model which is a clear example of an inaccessible LM that shows this ability. While RL4F is general enough to accommodate an ensemble of feedback generators, in this work we focus one single model as $\text{LM}_{\text{critique}}$ for simplicity.

In summary, this work²:

- Presents a reinforced critique generator which advances simple supervision in improving the end-task performance without retraining the downstream model.
- Demonstrates effectiveness of RL4F on three tasks: topic-based summarization, action planning and alphabetization (e.g. sorting a list of words alphabetically) with relative improvements up to 10%.
- Showcases that RL4F exhibits promising scaling properties and remains to be useful when applied iteratively.

2 Related Works

Past works differ to a large extent with respect to what they call human feedback and how they make use of it. In this section, after elucidating the use of human feedback in previous works, we briefly describe connections of RL4F to the parameter-efficient fine-tuning and discrete prompt learning literature.

²Code for the experiments is released under <https://github.com/fezyaakurek/r14f>.

2.1 What kind of feedback is used and where does it originate?

Human feedback on model predictions come in different flavors. The most notable ones include (1) binary feedback, e.g. thumbs up/down and pairwise comparisons (Ouyang et al., 2022; Bai et al., 2022; Gao et al., 2022), (2) natural language critiques (Tandon et al., 2022; Schick et al., 2022; Madaan et al., 2022; Saunders et al., 2022; Murty et al., 2022; Chen et al., 2023; Madaan et al., 2023) and (3) direct textual refinements to outcomes (Scheurer et al., 2022; Shi et al., 2022).

Bai et al. (2022) introduce what they call Reinforcement Learning from AI Feedback (RLAIF) in which they replace human preference labels with those of the model’s itself; the model is prompted to evaluate its own predictions in consideration of human values and preferences. In a similar vein, Gao et al. (2022) use accuracy for extractive question answering as a reward signal when fine-tuning their policy model.

In another thread, Schick et al. (2022) use comments from forums and Wikipedia edit histories as natural language feedback. Scheurer et al. (2022) and Shi et al. (2022) collect human natural language critiques and associated refinements. They then fine-tune the task model on the refinements. Our work is similar to these works in that we also use human-generated critiques in the first stage of our algorithm. Aside from human-written critiques, we additionally use synthetically generated critiques in the absence of the former.

2.2 How is feedback used?

An overwhelming majority of past work simply fine-tunes their task model using human feedback; whether it is a general purpose language model (Ouyang et al., 2022; Bai et al., 2022) or a task-specific model (Shi et al., 2022; Gao et al., 2022; Saunders et al., 2022; Scheurer et al., 2022; Schick et al., 2022). Tandon et al. (2022) differently fine-tunes a separate corrector model which takes in a *retrieved* critique utterance to correct initial outputs. Similarly, Madaan et al. (2022) retrieves from a memory of previous critiques to improve GPT-3 predictions via few-shot prompting.

Our work separates from existing work by focusing on critique generation and harnessing critiques that yield better final outcomes by LM_{task} . Similar to Schick et al. (2022), we effectively propose a multi-agent setup by disentangling critique gener-

ation and conditional refinement. Differently, we keep the latter model intact and only train the critique generator LM_{critique} via reinforcement learning. Moreover, we take a step forward by leveraging end task data for the first time and directly optimize the critique generation process to improve final task performance. In contrast to RLHF whose policy network (LM_{task}) is trained to maximize human alignment (Wiener, 1960), our policy network (LM_{critique}) is trained to bootstrap end-task success of LM_{task} . Our proposal RL4F is orthogonal to RLHF; in fact we use an RLHF fine-tuned checkpoint in our experiments. For further discussion, please refer to Fernandes et al. (2023) who catalogue different approaches on integrating natural language feedback to textual generations.

2.3 Adapters & Discrete Prompt Learning

A large body of existing work finds that parameter-efficient fine-tuning, often referred to as adapters (Pfeiffer et al., 2020) is as effective as full fine-tuning while being computationally cheap. RL4F can also be interpreted as an alternative “adapter” under the strict setting where only textual access to task model is available. Furthermore, our work can also be viewed from the perspective of learning discrete prompts for language models. Past work propose to generate knowledge pieces (Liu et al., 2022) or arbitrary textual snippets (Deng et al., 2022) which they append to the input via reinforcement learning. These works are different than ours in that their policy is conditioned solely on the input x whereas in our case we sample critiques of machine-generated predictions based on x and \hat{y} .

3 Background

The problem of learning from critiques entails two major challenges: (1) generating critiques and (2) the task of refining initial answers based on a critique. In our experiments (Section 6), GPT-3 responds very well to ground-truth critiques. This observation suggests that given quality critique, GPT-3 is indeed able to improve a potentially erroneous prediction for the better. Hence, in this study we focus our efforts on (1). Our ultimate goal is to reach, and eventually exceed, human-level critique performance using machines.

Following Saunders et al. (2022), we identify four primary functions towards studying the problem of learning with natural language critiques.

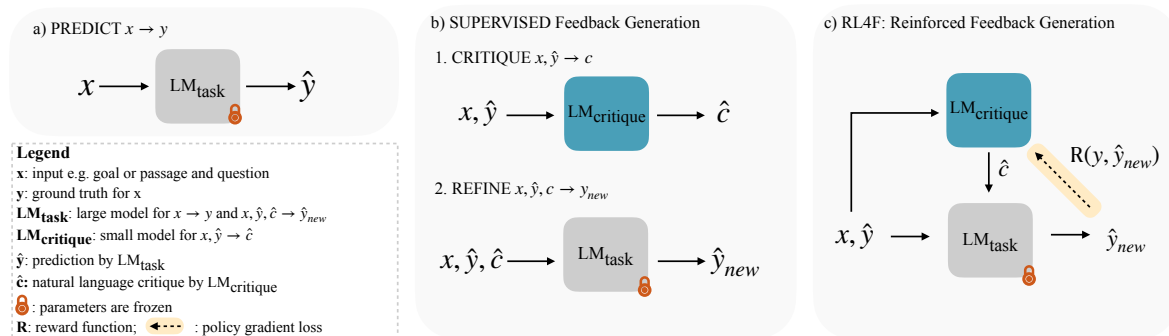


Figure 2: a) A downstream task model takes in an input (e.g. a passage and a question) and predicts the output (e.g. summary). b) Past work proposed using a supervised learning scheme (Saunders et al., 2022; Schick et al., 2022) or retrieval (Madaan et al., 2022) for critique generation (CRITIQUE) and refinement tasks (REFINE). In our setting, we only train $LM_{critique}$ and parameters of the task model are left unchanged. c) RL4F uses $LM_{critique}$ that was produced as a result of the training in part b. Using task data pairs (e.g. passages and summaries) we continue fine-tuning $LM_{critique}$ with policy gradient such that critiques steer LM_{task} to produce better outputs.

First is PREDICT: the base task of predicting without using critiques to model $x \rightarrow y$. As an example, if x is a passage, y is the summary (see Fig. 1). Moreover, we refer the task of learning to generate critiques $x, \hat{y} \rightarrow c$ where $\hat{y} \sim LM_{task}(x)$ as CRITIQUE. Lastly, we call the conditional refinement objective $x, \hat{y}, c \rightarrow y$ as REFINE and repairing an answer without a critique $x, \hat{y} \rightarrow y$ as DIRECTREFINE³. We use \hat{y} and \hat{c} notation to indicate an estimate of ground truth y , and similarly for c , from a respective model.

3.1 SUPERVISED: Supervised Learning for Critique Generation

We initialize $LM_{critique}$ to be a pretrained encoder-decoder-type model and fine-tune it to generate critiques satisfying the CRITIQUE objective $x, \hat{y} \rightarrow c$ using natural language critiques. Namely, if $LM_{critique}$ is parameterized by θ we maximize $\mathbb{E}[\log p_{\theta}(c|x, \hat{y})]$. We delegate PREDICT and REFINE tasks to GPT-3 via in-context learning. The procedure is depicted in Fig. 2a-b.

The main difference of our implementation of SUPERVISED to that of Saunders et al. (2022)’s is that we rely on separate models for CRITIQUE and the rest of the tasks while they train a single GPT-3-style model to collectively achieve PREDICT, CRITIQUE, REFINE and DIRECTREFINE; effectively merging $LM_{critique}$ and LM_{task} into a single model. While this may seem parameter-efficient, our version has a few key advantages. First, leaving any

³Additionally, Saunders et al. (2022) put forth the tasks of *critiqueability* to gauge whether an output requires a critique in the first place and *helpfulness* referring to if a sampled critique is useful. We let the critique generator model to implicitly take care of these tasks.

LM_{task} model intact (parameters frozen) enables us to work with models that are already-deployed as LM_{task} and those with expensive training and inference processes. Moreover, our approach refrains from disturbing overall integrity of a general-purpose language model by conditioning it to a specific task. Lastly, training $LM_{critique}$, which is multiple orders of magnitude smaller than GPT-3, is much more computationally efficient and therefore accessible to a broader range of users.

3.2 Direct-Refinement

Madaan et al. (2023); Chen et al. (2023) propose that using the critiques from the model itself via few-shot prompting results in improved performance. On the contrary, Saunders et al. (2022) and Bai et al. (2022) argue that direct refinement (as denoted with DIRECTREFINE in this work) i.e. the practice of prompting a language model without self-generated critiques to *directly* repair its own answers proves a stronger baseline, especially when the model size is large $>50B$. They hypothesize that this is primarily due to model’s initial answers getting increasingly more difficult to self-critique as the model size grows. In fact, both Saunders et al. (2022) and Bai et al. (2022) showed that their largest model achieves superior end-task performance when performing DIRECTREFINE than refining using self-generated critiques. Hence, we use Direct-Refinement as a baseline and describe how we implement it via in-context learning in Section 6 while providing further discussions in Appendix B.5.

4 RL4F: Reinforcement Learning for Feedback Generation

SUPERVISED is straightforward to implement but it does not make use of any final task data ($x \rightarrow y$) that is usually more abundant than natural language critiques. Moreover, it fails to provide ground for adaptation when the critiques in the train set are generally applicable but not entirely tailored to improving a target model. We describe RL4F where we follow supervised training with policy gradient learning using end-task data in order to generate critiques. We assume that the task model LM_{task} is already deployed and treat it as a fixed module. In all of our implementations we train the natural language critique generator LM_{critique} alone. In both SUPERVISED and RL4F, LM_{critique} takes in the input x and an initial prediction \hat{y} and produces a (natural language) critique \hat{c} :

$$LM_{\text{critique}}(x, \hat{y}) = \hat{c}. \quad (1)$$

Fig. 2c provides an illustration of RL4F. We implement LM_{task} as GPT-3 given its adaptability into new tasks using few-shot prompting. Our implementation which is primarily based on the RL4LMs library⁴ (Ramamurthy et al., 2022) will be publicly available.

Learning via Policy Gradient We warm-start RL4F by first fine-tuning LM_{critique} for CRITIQUE which we defined as the supervised objective of learning to generate natural language critiques c conditioned on x, \hat{y} . We continue fine-tuning the policy network (LM_{critique}) to maximize the reward using Proximal Policy Optimization (Schulman et al., 2017). We utilize the implementation of PPO provided by Ramamurthy et al. (2022) and refer the readers to the original work about the details for KL-regularized PPO objective. While any policy gradient approach could be used e.g. REINFORCE (Williams, 1992), our initial experiments showed that PPO works best in this setting.

Pseudocode for RL4F is provided in Algorithm 1 where we use two sets of in-context examples for prompting GPT-3. We define E to be a set of in-context-learning examples in the form of (x, y) to get GPT-3 solve PREDICT. Similarly, E^c contains in-context examples to prompt GPT-3 to fix an initial attempt \hat{y} into y conditioned on the natural language critique c which we termed as REFINE; $E^c = \{(x, \hat{y}, c, y)\}$. As per our reward function,

⁴<https://github.com/allenai/RL4LMs>

Algorithm 1 RL4F

Pseudocode of the algorithm used to train feedback model.

Input: Dataset $\mathcal{D} = \{(x^i, \hat{y}^i, y^i)\}_{i=1}^N$ of size N
Input: Initial $LM_{\text{critique}}, LM_{\text{task}}$
Input: In-context examples for refinement E^c
repeat
 Sample mini-batch $\mathcal{D}_m = \{(x^m, y^m, \hat{y}^m)\}_{m=1}^M \sim \mathcal{D}$
 Sample $\hat{c} \sim LM_{\text{critique}}(x, \hat{y})$ for \mathcal{D}_m in parallel
 Sample $\hat{y}_{new} \sim LM_{\text{task}}(E^c, x, \hat{y}, \hat{c})$ for \mathcal{D}_m
 Compute KL-regularized rewards R_t ▷ Eq. (2)
 Compute the advantage estimate \hat{A}_t
 Update the LM_{critique} by maximizing the PPO objective
until convergence and **return** LM_{critique}

we opt to use a lexical similarity metric ROUGE (1/2/L) (Lin, 2004) in Eq. (2) for planning and summarization datasets. Measuring ROUGE is computationally fast, making it easy to use in an online learning setting. Reward is only collected at a terminal stage i.e. either when the end of sentence token is produced or maximum number of tokens is reached.

$$R(\hat{y}, y) = \text{mean}(R1(\hat{y}, y), R2(\hat{y}, y), RL(\hat{y}, y)) \quad (2)$$

5 Datasets

5.1 Topic-Based Summarization

Saunders et al. (2022) crowd-sourced natural language critiques for topic-based summarization. The train, validation and test sets contain 14230, 1150 and 2658 tuples of (x, \hat{y}, \hat{c}) . The dataset provides multiple questions for a given passage each inquiring about a different aspect. Given a passage and question (x) multiple summaries are sampled from the model. Human annotators provide natural language critiques for the answers along with improved summaries. One example is provided in Fig. 1 and more are available in the Appendix (Table 8).

5.2 Interscript

Interscript (Tandon et al., 2021) is an action planning dataset for everyday tasks such as “put on a costume” or “play musical chairs”. Each goal x is associated with a sequence of ground truth actions y . Along with x, y pairs, it contains erroneous action plans \hat{y} and natural language critiques \hat{c} suggesting a fix. An example is provided in Fig. 1 for “put soap in dishwasher”. Other examples of critiques are “You need to have music to play musi-

cal chairs.” and “You need to pay for the costume before leaving the store”. More examples are available in the Appendix (see Table 6). Interscript represents a low-resource scenario: it contains 253, 45 and 169 examples for train, validation and test sets where each example contains 1-4 reference texts.

5.3 Synthetic Task: Alphabetization

We synthetically generate a task for alphabetically sorting a list of words with lengths ranging between 3-12. We use the lexicon #11 by [Keith Vertanen \(2018\)](#) which contains 43K unique English words. Given an unsorted list and a ground truth sorting of the list we identify 5 operations to sample a incorrect sorting of y denoted by \hat{y} and associated critique c articulating what is wrong about \hat{y} in natural language. One example is shown below:

x: mug, greek, book, house
y: book, greek, house, mug
 \hat{y} : book, greek, house
c: The word mug is missing.

The operations we use for distortion are REORDER, REPLACE, ADD, REPEAT and REMOVE (shown above). We also leave majority of sorted lists intact for which the ground truth critique is “The list is correctly sorted”. We use a total of 40K examples for warm-starting $LM_{critique}$ for the CRITIQUE objective and another 10K, 1K and 1K examples for PPO stage, for train, dev and test splits, respectively. Examples delineating other operations in action and corresponding natural language critiques are provided in Appendix A.

In alphabetization, we use Inverse-Levenstein distance for the reward function R as defined in Eq. (3) where $|\cdot|$ measures length of the list. Levenstein distance is a form of edit distance for single character edit operations such as removal, insertion and substitution in a string. We count word-level operations rather than character-level. Note that the higher inverse-Levenstein score of a predicted ordering, the closer it is to the alphabetically sorted version. The sorted list gets the maximum reward of 1.

$$R(\hat{y}, y) = 1 - \frac{\text{Levenstein}(\hat{y}, y)}{\max(|\hat{y}|, |y|)} \quad (3)$$

6 Experiments and Results

Our experiments are designed to test effectiveness of RL4F, along with other sources of critiques, in

both natural and controlled settings. In our evaluations, we test the usefulness of critiques by looking at the final task performance rather than evaluating generated critiques themselves; as multiple critiques may lead to the same correct answer.

Sampling Critiques We sample critiques from $LM_{critique}$ as in Eq. (1) by first concatenating the input and initial prediction. The specific input format for $LM_{critique}$ we use for Interscript is given below and the other two can be found in Appendix B.1. We initialize $LM_{critique}$ with pretrained T5-large which is a 0.77M parameter encoder-decoder type language model trained on large web text ([Raffel et al., 2020](#)).

Goal: {goal}
Steps: {initial_answer}

Downstream Predictions In our experiments, we consider GPT-3 as the LM_{task} model. GPT-3 can handle a wide range of tasks with prompting—using a handful of task examples in the input and without requiring task-specific fine-tuning ([Brown et al., 2020](#)). GPT-3 is not only able to tackle numerous tasks conveniently but also can refine initial predictions when given a natural language critique ([Madaan et al., 2022](#)). Since our setting requires LM_{task} model to be able to model both the main task objective $x \rightarrow y$ and the refinement objective $x, \hat{y}, c \rightarrow y$, GPT-3 is a suitable candidate that can adapt to both, using few-shot exemplars. The prompt template we use for the latter is shown in Fig. 3 where we provide the model with an initial attempt to the question `initial_answer` and re-sample a revised prediction conditioned on the question and critique for the summarization task. We use code-davinci-002 checkpoint via OpenAI API⁵ and 3, 1 and 6 hand-written in-context examples for planning, summarization and alphabetizations tasks, respectively as we exhaust the 4096 token input limit.

In action planning, instead of resampling entire plans, we prompt GPT-3 to produce an edit operation on the initial plan \hat{y} . The set of edit operations identified in the original dataset are *Insert*, *Remove* and *Reorder* where each critique comes with a corresponding edit operation. Note that, these operations can algorithmically be applied to \hat{y} . While *Reorder* and *Remove* are expected to refer to existing steps in \hat{y} , we expect *Insert* to introduce a

⁵We find that only the largest 175B parameter GPT-3 can handle REFINE.

Source of Critiques	Interscript			Topic-Based Summarization		
	BLEURT \uparrow	BERTScore \uparrow	R1/R2/RL \uparrow	BLEURT \uparrow	BERTScore \uparrow	R1/R2/RL \uparrow
Direct-Refinement	-1.07	86.97	15.8 / 0.9 / 15.5	0.09	93.1	54.3/46.0/50.9
SUPERVISED	-1.02	86.99	19.4 / 0.5 / 18.5	0.06	92.9	53.2/46.4/50.7
MemPrompt	-1.18	87.45	16.9 / 1.9 / 16.7	0.09	91.9	48.8 / 40.4 / 45.6
RL4F (Ours)	-0.92	87.23	22.1 / 0.9 / 21.3	0.10	93.6	55.1/48.2/52.6
<i>With gold feedback</i>	-0.69	89.56	40.7 / 6.8 / 39.1	0.22	94.2	58.3/50.3/55.8

Table 1: Results for action sequence generation with Interscript (Tandon et al., 2021) and topic-based summarization by Saunders et al. (2022). We evaluate the performance of different sources for natural language critiques in steering LM_{task} to improve its predictions. Best scores in each column are made bold. We compare our method, RL4F, to three strong baselines and human-generated critiques. Self-Refinement prompts GPT-3 to self-repair its answer. MemPrompt uses memory to store human-generated critiques to previous outputs (Madaan et al., 2022). ROUGE and BERTScore are out of 100 while BLEURT can be negative or positive and should be used in comparing different methods.

Source of Critiques	Exact Match	Inverse Levenstein
Initial Outputs (\hat{y})	63.7	0.91
Fine-tuning davinci	55.3	0.89
MemPrompt	57.8	0.89
Direct-Refinement	65.9	0.92
SUPERVISED	38.9	0.82
RL4F	66.1	0.92
<i>With gold feedback</i>	75.9	0.94

Table 2: Results for alphabetization. Best scores are highlighted. Initial Outputs are obtained from GPT-3 (code-davinci-002) via in-context learning. SUPERVISED critiques misguides GPT-3, hurting its initial performance, as with MemPrompt. RL4F improves over the performance of SUPERVISED model by 27 absolute points. Self-Refinement around the same as RL4F. In Fig. 5, we further discuss advantages of RL4F over Self-Refinement when we sample and refine iteratively.

novel action. Hence, we stick with a generic lexical similarity metric in calculating reward (Eq. (2)) for this task. In summarization, we compare human-written summaries with the repaired model summaries.

Baselines We compare effectiveness of RL4F to SUPERVISED which is described in Section 3.1. This is the closest baseline to the approach by Saunders et al. (2022) and Schick et al. (2022) that abides by our condition that LM_{task} should remain unchanged. We use the same set of initial predictions \hat{y} when comparing different critique generators.

In addition to SUPERVISED, we use a simple Direct-Refinement baseline where we ask LM_{task} to revise the initial prediction given a fixed critique “Improve the answer.” (DIRECTREFINE). The prompt template is otherwise the same as in other methods. We configure our in-context examples to show that not all \hat{y} need to be repaired. Hence, LM_{task} is free to update the prediction or leave

```

Edit the below summary of the passage taking into
account the remarks in the feedback.
---
{passage}
Question: {question}
Answer: {initial_answer}
Question: {question} {critique}
Answer: {new_answer}
---
{passage}
Question: ...

```

Figure 3: Prompt template for topic-based summarization. We ask GPT-3 to refine the initial prediction by using critique.

it as is when it is correct. Despite its simplicity, Direct-Refinement has been established as a strong baseline (Saunders et al., 2022).

Moreover, we compare to MemPrompt (Madaan et al., 2022). In their work, authors study a setup where LM_{task} generates an *understanding* along with the target output. For example, given a question “What sounds like good?”, the model generates an understanding of the question “The question is asking for homonym.” before saying “wood”. In their critique retriever, they train a mapping to model x into an understanding u . However *understanding* is redundant in particular tasks e.g. summarization where the question is no different than u , thus throughout our experiments, we replace the learned retriever in MemPrompt with BM25 (Harter, 1975).

Lastly, we use human-written critiques (gold feedback) for REFINE in getting LM_{task} to repair outputs and report this as an upperbound.

6.1 Planning and Summarization

Our main results for Interscript and topic-based summarization are provided in Table 1. Given the

free-from nature of the outputs, we evaluate planning and summarization tasks using text similarity metrics to capture semantic and lexical similarities. We utilize learned metrics such as BLEURT (Selam et al., 2020) and BERTScore (Zhang* et al., 2020) along with ROUGE (Lin, 2004). We compare the performance achieved by using different sources of critiques to that of human-written critiques. Across all metrics, RL4F yields one of the closest outcomes to human-written critiques.⁶

6.2 Alphabetization

We initialize our $LM_{critique}$ using the synthetic critiques as described Section 5.3. Our results are provided in Table 2. For alphabetization we compute exact match and inverse-Levenshtein scores as defined in Eq. (3). As an additional baseline, we fine-tune davinci (Brown et al., 2020) on the same train set as our RL4F.

Because of the synthetic procedure to create x, \hat{y}, c triplets, the generated \hat{y} as well as c do not necessarily reflect the kinds of errors that LM_{task} would do. We observe this in the scores of SUPERVISED which fails to improve upon initial outputs. Nevertheless, RL4F procedure helps the policy network to capture a useful distribution of critiques, improving over SUPERVISED by more than 27 absolute points. In this simple task, Direct-Refinement prompt also yields a competitive performance. Compared to full fine-tuning, we observe that despite training substantially fewer parameters RL4F achieves a significantly better accuracy. For a comparison to concurrent work Madaan et al. (2023), please refer to the appendix.

7 Analysis

Scaling Properties While we use T5-large as our main model for all of our experiments to initialize $LM_{critique}$, we inquire about different model sizes. In Fig. 4, we consider three different model sizes to tackle Interscript ranging from 60M to 770M parameters. On the y-axis we provide average of three ROUGE scores for the generated plans. RL4F greatly benefits from an increase in the model size where a similar trend in SUPERVISED is non-existent.

⁶We have identified a handful of examples where a pair of train and test examples differs by only a single concept e.g. all occurrences of “noodle” in the train sample was replaced with “food” to produce the test sample. The goal and steps are the same otherwise. MemPrompt does exceedingly well on these 7 cases, hence performing occasionally higher, yet fails in the rest of the test/val examples.

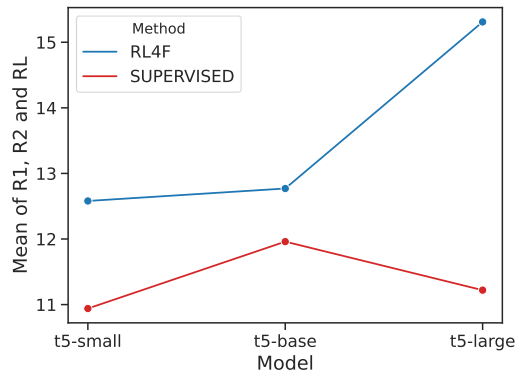


Figure 4: Scaling properties of SUPERVISED and RL4F on Interscript. We observe that RL4F greatly benefits from an increase in the number of parameters.

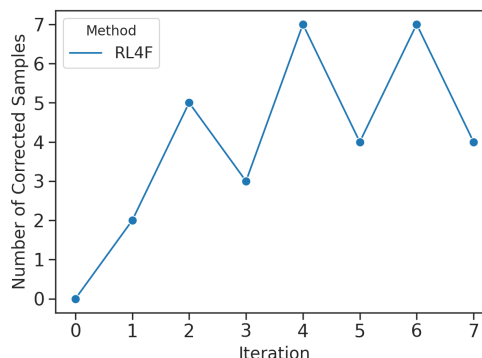


Figure 5: We apply REFINE multiple times given critiques from $LM_{critique}$ (RL4F) on alphabetization task. RL4F leads to a handful of more corrections when used iteratively.

Semantic Drift In goal-oriented training, semantic drift occurs when the strings produced by the policy begin diverging from initial language (Lee et al., 2019; Blank, 1999). Although, RL4F does not guarantee that $\hat{c} \sim LM_{critique}$ will be natural language, we find minimal sign of semantic drift in the sampled critiques with respect to fluency and naturalness. In most cases, generated critiques are intelligible. We speculate that may be due to GPT-3 responding to natural language best than gibberish, though future work should look closely into this to make a more conclusive argument. We provide sample predictions from both models in Appendix C for all three tasks.

Iterative Improvement In Section 6, we provide results with applying only one round of critiques in alphabetization. Past work advocated for iterative editing (Reid and Neubig, 2022; Faltings et al., 2021) as opposed to one-shot editing. In Fig. 5, we sample and apply critiques from $LM_{critique}$ to \hat{y} ’s iteratively and see if the number of correctly

sorted lists increase or decrease. Note that critiques may also lead to deteriorating performance as we are not eliminating the solved examples *and* it is at $LM_{critique}$'s discretion to declare a solution as correct e.g. by saying “*This list is correctly sorted.*”. In fact, when we ask GPT-3 to simply improve its predictions iteratively (via Direct-Refinement as described in Section 3.2), it occasionally scrambles an already correct ordering while not scoring any new points. In contrast, RL4F leads to up to 7 more corrections (see Fig. 5).

8 Conclusion

We have described a collaborative framework involving two language models where one model, the critique generator, is trained to improve the performance of the task model. We train the former via policy gradient learning while treating the task model as a black-box. We show that RL4F leads to superior final performance across three domains compared to other strong baselines without resulting as the critiques remain fluent and natural. Future work might focus on generalizing the critique generator into a mixture of experts allowing humans and other models to contribute to critiquing procedure.

9 Limitations

RL4F is primarily targeted at improving final performance. While we have found that the critiques learned by RL4F remain natural, we do not introduce any explicit restraints preventing semantic drift. As though it may raise end-task performance, semantic drift would also hinder interpretability. Future work might study datasets that are not covered by this dataset and quantify semantic drift along with proposing measures to prevent it, as necessary. Moreover, this work does not provide an explicit mechanism to incorporate new critique labels that might become available in the future nor it identifies a framework that could combine critiques from multiple experts such humans and other machines. Lastly, we limit our analysis to GPT-3 and focus on a scenario where it is inefficient or impossible to train the task model while this may be a conservative assumption for other settings.

Acknowledgement

We thank Aman Madaan for sharing his insights throughout this project. We also thank Anna

Ivanova, Jacob Andreas, Zilu Tang, Shashank Gupta and Ashish Sabharwal for their valuable feedback on earlier drafts of this paper. Finally, we thank Rajkumar Ramamurthy and Prithviraj Ammanabrolu for helpful discussions on using their RL4LMs repository which facilitated the experiments of this work.

Afra Feyza Akyürek is supported in part by the U.S. NSF grant 1838193 and DARPA HR001118S0044 (the LwLL program). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA and the U.S. Government. At MIT, Ekin Akyürek is supported by an MIT-Amazon ScienceHub fellowship and by the MIT-IBM Watson AI Lab.

References

- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.
- Andreas Blank. 1999. Why do new meanings occur? a cognitive typology of the motivations for lexical semantic change andreas blank. *Cognitive Linguistics Research*, page 61.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.
- Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.
- David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A Saurous, Jascha Sohl-Dickstein, et al. 2022. Language model cascades. *arXiv preprint arXiv:2207.10342*.
- Felix Faltings, Michel Galley, Gerold Hintz, Chris Brockett, Chris Quirk, Jianfeng Gao, and Bill Dolan. 2021. [Text editing by command](#). In *Proceedings of*

- the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5259–5274, Online. Association for Computational Linguistics.
- Patrick Fernandes, Aman Madaan, Emmy Liu, António Farinhas, Pedro Henrique Martins, Amanda Bertsch, José GC de Souza, Shuyan Zhou, Tongshuang Wu, Graham Neubig, et al. 2023. Bridging the gap: A survey on integrating (human) feedback for natural language generation. *arXiv preprint arXiv:2305.00955*.
- Ge Gao, Eunsol Choi, and Yoav Artzi. 2022. Simulating bandit learning from user feedback for extractive question answering. *arXiv preprint arXiv:2203.10079*.
- Stephen P Harter. 1975. A probabilistic approach to automatic keyword indexing. part i. on the distribution of specialty words in a technical literature. *Journal of the american society for information science*, 26(4):197–206.
- Keith Vertanen. 2018. [Big english word lists](#).
- Jason Lee, Kyunghyun Cho, and Douwe Kiela. 2019. Countering language drift via visual grounding. *arXiv preprint arXiv:1909.04499*.
- Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. [Corpora generation for grammatical error correction](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3291–3301, Minneapolis, Minnesota. Association for Computational Linguistics.
- Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.
- Jiacheng Liu, Skyler Hallinan, Ximing Lu, Pengfei He, Sean Welleck, Hannaneh Hajishirzi, and Yejin Choi. 2022. Rainier: Reinforced knowledge introspector for commonsense question answering. *arXiv preprint arXiv:2210.03078*.
- Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve gpt-3 after deployment. *arXiv preprint arXiv:2201.06009*.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhunoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.
- Jonathan Mallinson, Aliaksei Severyn, Eric Malmi, and Guillermo Garrido. 2020. [FELIX: Flexible text editing through tagging and insertion](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1244–1255, Online. Association for Computational Linguistics.
- Eric Malmi, Yue Dong, Jonathan Mallinson, Aleksandr Chuklin, Jakub Adamek, Daniil Mirylenka, Felix Stahlberg, Sebastian Krause, Shankar Kumar, and Aliaksei Severyn. 2022. [Text generation with text-editing models](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*, pages 1–7, Seattle, United States. Association for Computational Linguistics.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022a. [Fast model editing at scale](#). In *International Conference on Learning Representations*.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.
- Shikhar Murty, Christopher D Manning, Scott Lundberg, and Marco Tulio Ribeiro. 2022. Fixing model bugs with natural language patches. *arXiv preprint arXiv:2211.03318*.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.
- Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. [Adapterhub: A framework for adapting transformers](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020): Systems Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.
- Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2022. Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*.
- Machel Reid and Graham Neubig. 2022. Learning to model editing processes. *arXiv preprint arXiv:2205.12374*.
- Joana Ribeiro, Shashi Narayan, Shay B. Cohen, and Xavier Carreras. 2018. [Local string transduction as sequence labeling](#). In *Proceedings of the 27th International Conference on Computational Linguistics*,

- pages 1360–1371, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*.
- Jérémy Scheurer, Jon Ander Campos, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2022. Training language models with natural language feedback. *arXiv preprint arXiv:2204.14146*.
- Timo Schick, Jane Dwivedi-Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2022. Peer: A collaborative language model. *arXiv preprint arXiv:2208.11663*.
- Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in NLP. *Transactions of the Association for Computational Linguistics*, 9:1408–1424.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.
- Weiyang Shi, Emily Dinan, Kurt Shuster, Jason Weston, and Jing Xu. 2022. When life gives you lemons, make cherryade: Converting feedback from bad responses into good labels. *arXiv preprint arXiv:2210.15893*.
- Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1630–1640, Florence, Italy. Association for Computational Linguistics.
- Niket Tandon, Aman Madaan, Peter Clark, Keisuke Sakaguchi, and Yiming Yang. 2021. Interscript: A dataset for interactive learning of scripts through error feedback. *arXiv preprint arXiv:2112.07867*.
- Niket Tandon, Aman Madaan, Peter Clark, and Yiming Yang. 2022. Learning to repair: Repairing model output errors after deployment using a dynamic memory of feedback. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 339–352, Seattle, United States. Association for Computational Linguistics.
- Norbert Wiener. 1960. Some moral and technical consequences of automation: As machines learn they may develop unforeseen strategies at rates that baffle their programmers. *Science*, 131(3410):1355–1358.
- Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.
- Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Dataset Processing

Action Planning Interscript is larger but we are only using a subset, removing distractors. The scripts used for data cleaning will be released along with the codebase.

Alphabetization We sample initial predictions from GPT-3 for alphabetization some of which comprise multiple distortions simultaneously, yet we use one-step distortions to warm-start LM_{critique}.

Given the following a pair of unsorted and sorted word lists e.g.

x: mug, greek, book, house
y: book, greek, house, mug

below are the operations we used to create our data:

REORDER

\hat{y} : book, house, greek, mug
c: The word greek is placed in an incorrect position.

REPLACE

\hat{y} : book, greek, house, mud
c: The word mug is replaced with mud

REMOVE

\hat{y} : book, greek, mug
c: The word house is missing

REPEAT

\hat{y} : book, house, greek, house, mug
c: The word house is repeated

ADD

\hat{y} : book, hair, greek, house, mug
c: The word hair is not in the original list

NOTHING

\hat{y} : book, greek, house, mug
 c: The list is correctly sorted.

B Experiment Details

We use code-davinci-002 as GPT-3 unless otherwise specified. We compute ROUGE implementation in the datasets library and set use_stemmer=True for summarization and Interscript.

B.1 Data Formats

We use the following input formats for $LM_{critique}$:

- **Summarization:**

```
{passage}
Question: {question}
Answer: {initial_answer}
```

- **Planning:**

```
Goal: {goal}    Steps: {steps}
```

- **Alphabetization:**

```
{unsorted_list} ||| {initial_answer}
```

We train separate models for each of the datasets and evaluate individually. We use T5-large provided by transformers library.

B.2 Prompts for GPT-3

We provide prompt templates used for Alphabetization and Interscript when prompting GPT-3 for REFINE. Template for summarization is provided in the main text.

- **Interscript:**

```
Goal: {goal}
Steps: {steps}
Feedback: {critique}
Edit:
```

- **Alphabetization:**

```
{unsorted_list} ||| {initial_answer}
Feedback: {critique}
Edit:
```

In Direct-Refinement, templates remain the same and critique’s are replaced with “Improve the answer.”. Exact prompt exemplars will be made available in the released code repository.

Model Params	value
supervised	batch size: 8 epochs: 10 learning rate: 0.00001 learning rate scheduler: cosine weight decay: 0.01
supervised+ppo	steps per update: 240 total number of steps: 96,000 batch size: 24 epochs per update: 5 learning rate: 0.000001 entropy coefficient: 0.001 initial kl coeff: 0.00001 target kl: 3 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 value function coeff: 0.5 rollouts top k: 100
decoding	sampling: True temperature: 0.7 min length: 5 max new tokens: 20
tokenizer	padding side: left truncation side: right max length: 512

Table 3: List of hyperparameters for Alphabetization.

B.3 Standard Deviations

Standard deviations of R1/R2/RL scores across 5 runs in Interscript are 1.4/0.1/1.1 for SUPERVISED and 0.5/0.4/0.5 for RL4F.

B.4 Hyperparameters

In all of our experiments we use temperature 0 for prompting GPT-3 except when sampling initial predictions for alphabetization we set it to 0.5. We provide hyperparameters for RL4LMs (Ramamurthy et al., 2022) in Table 3, Table 4 and Table 5.

B.5 Results for *Self-Refine* (Madaan et al., 2023) on Alphabetization

Madaan et al. (2023) and Chen et al. (2023) propose that self-generated critiques (sampling critiques simply via few-shot prompting) is useful for a range of tasks. We examine if self-generated critiques are more useful for Alphabetization task than other techniques proposed in Table 2. Doing so, we curate a few-shot prompt:

Below is a given list of words which are supposed to be sorted in alphabetical order.

Model Params	value
supervised	batch size: 5 epochs: 5 learning rate: 0.00001 learning rate scheduler: cosine weight decay: 0.01
supervised+ppo	steps per update: 256 total number of epochs: 256,000 batch size: 8 epochs per update: 5 learning rate: 0.0000005 entropy coefficient: 0.001 initial kl coeff: 0.01 target kl: 2 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 value function coeff: 0.5 rollouts temperature: 0.7
decoding	sampling: True temperature: 0.3 min length: 15 max new tokens: 50 repetition penalty: 0.2
tokenizer	padding side: left truncation side: left max length: 512

Table 4: List of hyperparameters for Interscript.

Model Params	value
supervised	batch size: 4 epochs: 7 learning rate: 0.00001 learning rate scheduler: cosine weight decay: 0.01
supervised+ppo	steps per update: 1024 total number of epochs: 143,360 batch size: 4 epochs per update: 3 learning rate: 0.0000001 entropy coefficient: 0.001 initial kl coeff: 0.01 target kl: 2 rollouts temperature: 0.7
decoding	sampling: True temperature: 0.7 min length: 20 max new tokens: 150 repetition penalty: 0.2
tokenizer	padding side: right truncation side: right max length: 1024

Table 5: List of hyperparameters for Topic-Based Summarization dataset by [Saunders et al. \(2022\)](#).

Describe what is wrong in the provided ordering.

Ordering: quirky whimsical bubbly joyous
delightful melodic glimmering vivacious
radiant lively zestful spontaneous
Feedback: Whimsical should come in the end.
Delightful should come before joyous.

Ordering: airy amiable animated ardent
astute beaming blithe brilliant
Feedback: This listed is correctly sorted.

Ordering: curious sprightly vivacious
tenacious passionate vivacious
Feedback: The list contains duplicates
and passionate should come before
sprightly.

Ordering: {ordering}
Feedback:

After sampling critiques using the above prompt, we follow the same steps described in Section 6 for refinement. We obtain 21.6% exact match accuracy using these critiques which is a significant drop from code-davinci-002’s initial performance. When we sample critiques and refinements from text-davinci-002, the exact match score increases notably (to 58.6%) while still hurting the initial accuracy of 63.6%. Self-Refine may improve over initial performance when critiques are sampled from more capable models such as text-davinci-003 ([Madaan et al., 2023](#)). Nonetheless, having initialized as T5-large, RL4F’s critique model already produces useful feedback despite being significantly smaller than text-davinci-003.

C Sample Predictions

Sample predictions for all three tasks are provided in Table 6, Table 7 and Table 8. A manual examination of the generated critiques revealed that close to 100% of the critiques generated for alphabetization and action planning are grammatical. However, critiques for the topic-based-summarization task often involve repetitions or generic calls to fix or improve the answer e.g. "The answer is not adequate. The answer is wrong, it is wrong and should be fixed."

Input	SUPERVISED Critique	Edit w/ SUPERVISED	RL4F Critique	Edit w/ RL4F
Goal: go to the hardware store Steps: 1. pay for the items 2. decided to go to the hardware store 3. pack items in a bag 4. create a shopping list 5. go to the grocery store 6. drive to the store 7. walk around the store 8. put items in cart 9. pay for the groceries	You can't pay for groceries if you don't pay for the groceries. [Repetition]	Insert 'pay for the groceries' before 9 [Repeated nodes due to critique.]	The plan tells you to pay for the items before you go to the hardware store. [Correctly points out an error.]	Reorder the edge between 1 and 2 [GPT-3 correctly identifies the node to be replaced, yet the new location is not correct.]
Goal: ride the bus Steps: 1. decided to ride the bus 2. put on shoes and clothing 3. walk to the front door 4. grab the handle 5. open the door 6. walk out of the house 7. lock the door 8. ride the bus	You have to be in the house before you can get out. You can't get out of the house until you're in the house. [Logically sound, but unhelpful critique.]	Reorder the edge between 5 and 6 [GPT-3 fixes what it thinks the critique is saying.]	You have to be at the bus stop before you can ride it. [Points out an helpful correction.]	Insert 'walk to the bus stop' after 8 [A missing node is added to an incorrect place.]
Goal: turn on computer Steps: 1. sit down at computer 2. pull the chair closer to the computer 3. look for the power switch on the computer 4. check the power connection 5. move a finger to the power switch 6. press the power switch on the computer 7. relax until the computer boots up 8. turn on computer	You can't press the power switch on the computer if you don't have the power switch. [Unhelpful critique.]	Insert node 'find the power switch' before 6 [Critique results in incorrect edit.]	You have to press the power switch to turn on the computer. You have to do this before you can relax. [Points out what's already correct in the plan.]	Reorder the edge between 6 and 7 [Critique results in incorrect edit.]
Goal: pick up large kettle from counter Steps: 1. make sure ingredients are all ready 2. look for the pot 3. extend hand out 4. reach for the pot 5. grip the pot 6. lift the pot 7. pick up pot from counter	You need to reach for the pot before you can reach for it. [Incorrect critique involving repetition.]	Reorder the edge between 3 and 4 [Critique results in an unnecessary edit.]	Plan tells you to reach for the kettle before you reach for it. [Points out a redundancy in the plan.]	Remove node 3 [Removes unnecessary node.]

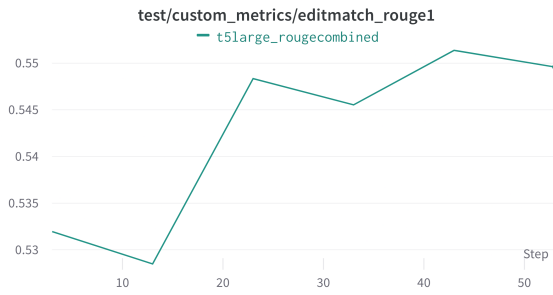
Table 6: Samples for Interscript (Tandon et al., 2021). The authors' comments evaluating the generations are given in brackets. Red comments indicate what went wrong and blue means correct. Refinements use the corresponding critiques.

Input	SUPERVISED Critique	SUPERVISED Refinement	RL4F Critique	RL4F Refinement
accomplishments catwalk accessory incidental sixty gad remission melted juices fellowship unto dredge accessory accomplishments catwalk dredge fellowship gad incidental juices melted remission sixty unto	The word melted is placed in an incorrect position. [The word melted is in the correct position.]	accessory accomplishments catwalk dredge fellowship gad incidental melted juices remission sixty unto	The list is correctly sorted. [Correct.]	accessory accomplishments catwalk dredge fellowship gad incidental juices melted remission sixty unto [Correct.]
inseparable snapping astrology advertising unjust indies dismantle bangkok boosters brasil businessmen advertising astrology bangkok brasil businessmen boosters dismantle indies inseparable injustice snapping	The word unjust is missing. [Correctly points out that unjust is missing.]	advertising astrology bangkok brasil businessmen boosters dismantle indies injustice inseparable snapping [GPT-3 fails to add unjust; keeps the word injustice which was not in the original list.]	The word unjust is replaced with injustice. [Correctly indicates unjust is replaced with injustice]	advertising astrology bangkok brasil businessmen boosters dismantle indies inseparable injustice snapping [GPT-3 fails to replace injustice with unjust.]

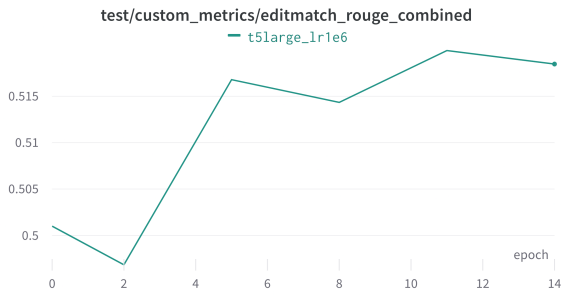
Table 7: Samples for Alphabetization. Unsorted list and GPT-3's attempt in sorting them are separated with ||| in the input.

Input	SUPERVISED Critique	SUPERVISED Refinement	RL4F Critique	RL4F Refinement
<p>Jeffery was sitting in his statistics class nodding away as the information was simply passing by him. He could hardly wait to get out of class. Mondays, and Fridays were fine because there was a small chance of him seeing her. However, on Wednesdays she was always on time. He could see it now walking out from his this room strolling down Taylor Mall.[...]</p>	<p>The response does not mention that Jeffery’s friend is killed. The response only mentions that he is saved. [No mention of murder in the story.]</p>	<p>Summer is a student in Jeffery’s statistics class. She is a beautiful girl with long dark hair and a sunflower dress. Every Wednesday at 12:30, she waits for Jeffery to come out of class so she can see his smiling face. However, she has never had the courage to ask him to a date. [GPT-3 ignores the critique and leaves the answer as is. It’s Jeffery who is interested in asking Summer out not the other way around.]</p>	<p>The answer doesn’t mention Jeffery asking her to a date. It should mention that he is not sure how to ask her to a date. [This is an important detail that’s is confused by the model’s initial prediction.]</p>	<p>Summer is a student in Jeffery’s statistics class. She is a beautiful girl with long dark hair and a sunflower dress. Every Wednesday at 12:30, she waits for Jeffery to come out of class so she can see his smiling face. However, she has never had the courage to ask him to a date. Jeffery is not sure how to ask her to a date. [GPT-3 successfully adds that it’s Jeffery who is into Summer, yet it fails to remove the previous incorrect sentence suggesting otherwise.]</p>
<p>Gary left the house at six am on Wednesday; headed out to endure another long workday. After he walked through the yard to the driveway, he made sure to latch the gate securely, because that’s what Sarah liked. He smiled as he did so, and thought of how earlier, he had closed the shower curtain all the way (another Sarah preference), and gently lowered the toilet seat. [...] Summarize how Gary feels about his life. Answer: Gary feels that his life is boring and that he is stuck in a rut. He feels that he is a slave to his wife and his job. He wants to be free of his responsibilities and wants to be a bartender.</p>	<p>The answer should mention that Gary gets a friend for dinner. There are other examples of a friendship that are mentioned in the text. [The feedback is not helpful with respect to the question which asks how Gary is feeling about life.]</p>	<p>Gary feels that his life is boring and that he is stuck in a rut. He feels that he is a slave to his wife and his job. He wants to be free of his responsibilities and wants to be a bartender. However, he gets a friend for dinner and there are other examples of a friendship that are mentioned in the text. [GPT-3 appends the irrelevant details suggested by the critique.]</p>	<p>It’s omitted that he is the one who is angry. He has a son, Steven. [Critique suggests a detail about his anger.]</p>	<p>Gary feels that his life is boring and that he is stuck in a rut. He feels that he is a slave to his wife and his job. He wants to be free of his responsibilities and wants to be a bartender. He is angry at his wife and has a son, Steven. [Having seen the critique finds out that Gary’s anger is actually towards his wife and improves its answer accordingly.]</p>
<p>SR-25 pattern magazine An SR-25, AR-10 or LR-308 PATTERN MAGAZINE is a type of detachable firearm magazine based on the original Armalite AR-10 “waffle” magazine design introduced in the late 1950s. [...] What does the text say about the SR-25, AR-10, or LR-308 Pattern Magazine? Answer: A detachable firearm magazine based on the original Armalite AR-10 ‘waffle’ magazine design introduced in the late 1950s, the design has been used on many AR-10 derivatives. It was reintroduced in 2012 with their new AR-10A model.</p>	<p>The response includes inaccurate information. Firstly, a reference to the SR-25 pattern is omitted. [The response does not contain inaccurate information and AR10A indeed contains SR25 pattern.]</p>	<p>A detachable firearm magazine based on the original Armalite AR-10 ‘waffle’ magazine design introduced in the late 1950s, the design has been used on many AR-10 derivatives. It was reintroduced in 2012 with their new AR-10A model. [GPT-3 repeats the initial answer.]</p>	<p>The response is missing several important information about the magazine. The first sentence should be more specific. [Critique calls for a more detailed summary.]</p>	<p>A detachable firearm magazine based on the original Armalite AR-10 ‘waffle’ magazine design introduced in the late 1950s, the design has been used on many AR-10 derivatives. It was reintroduced in 2012 with their new AR-10A model. The magazine is used in many AR-10 derivatives such as the Knight’s Armament SR-25 and DPMS Panther LR-308, as well as bolt action rifles such as the Mossberg MVP.308, Ruger Precision Rifle and Q FIX. Aftermarket magazines are produced by Magpul, Lancer Systems, and others. [More comprehensive summary of magazine designs.]</p>

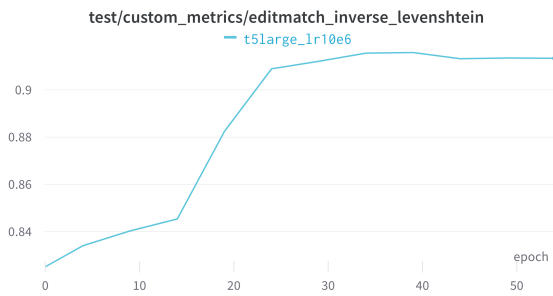
Table 8: Samples for Topic-Based Summarization by Saunders et al. (2022). Passages are truncated.



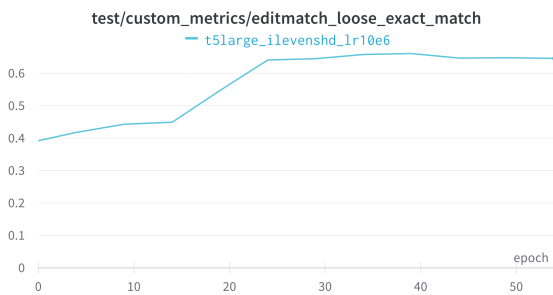
(a) ROUGE for topic-based summarization.



(b) Mean ROUGE for Interscript.



(c) Inverse Levenshtein distance for alphabetization.



(d) Exact match for alphabetization.

Figure 6: As RL4F is trained, we track how evaluation metrics evolve for dev and test sets. Here, we display the results assessing the revised outputs conditioned on the critiques for test samples.

D Learning Curves for Reinforcement Learning

In Fig. 6, we provide how evaluation metrics progress as $LM_{critique}$ in RL4F is trained.

ACL 2023 Responsible NLP Checklist

A For every submission:

- A1. Did you describe the limitations of your work?
9
- A2. Did you discuss any potential risks of your work?
Left blank.
- A3. Do the abstract and introduction summarize the paper’s main claims?
Left blank.
- A4. Have you used AI writing assistants when working on this paper?
Left blank.

B Did you use or create scientific artifacts?

6

- B1. Did you cite the creators of artifacts you used?
Left blank.
- B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Datasets we used are made publicly available by the authors for scientific use. Interscript is available via Apache 2.0.
- B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Left blank.
- B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Left blank.
- B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Left blank.
- B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C Did you run computational experiments?

6

- C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
6, Appendix B

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

- C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?

6, Appendix B

- C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?

6

- C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?

6, Appendix

D Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

- D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?

No response.

- D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?

No response.

- D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?

No response.

- D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?

No response.

- D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?

No response.