Generating Sequences by Learning to Self-Correct

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen,
Daniel Khashabi, Yejin Choi

Presenter: Nirav Diwan

Overview

1) Motivation
2) Problem Statement
3) Literature Review
4) Intuition
5) Method
6) Evaluation
7) Strengths
8) Possible Follow-up Work

Motivation

Language Models (rarely) get things right on the first try!

“Lets’ try this”

“It failed :(“

“What should
I change?”

Problem Statement

We want to learn from feedback.

How can we improve LM output for our task in a systematic way?

1) Systematic Way - Learning through feedback
2) Improvement - Measurable Change in Performance
3) For our Task - Task specific

Related Work

1) Rationale Generation - Ask a model to reason on its answer and use it as feedback to update the model
2) Denoising Ground Truth - Masked Language Modelling
3) Supervised edits - Train a model to improve based on wikipedia edits

Most methods -

1) Require large amount of data (usually supervised)
2) Updates all the parameters of large models (expensive)
3) Performance limited to specific tasks

Intuition

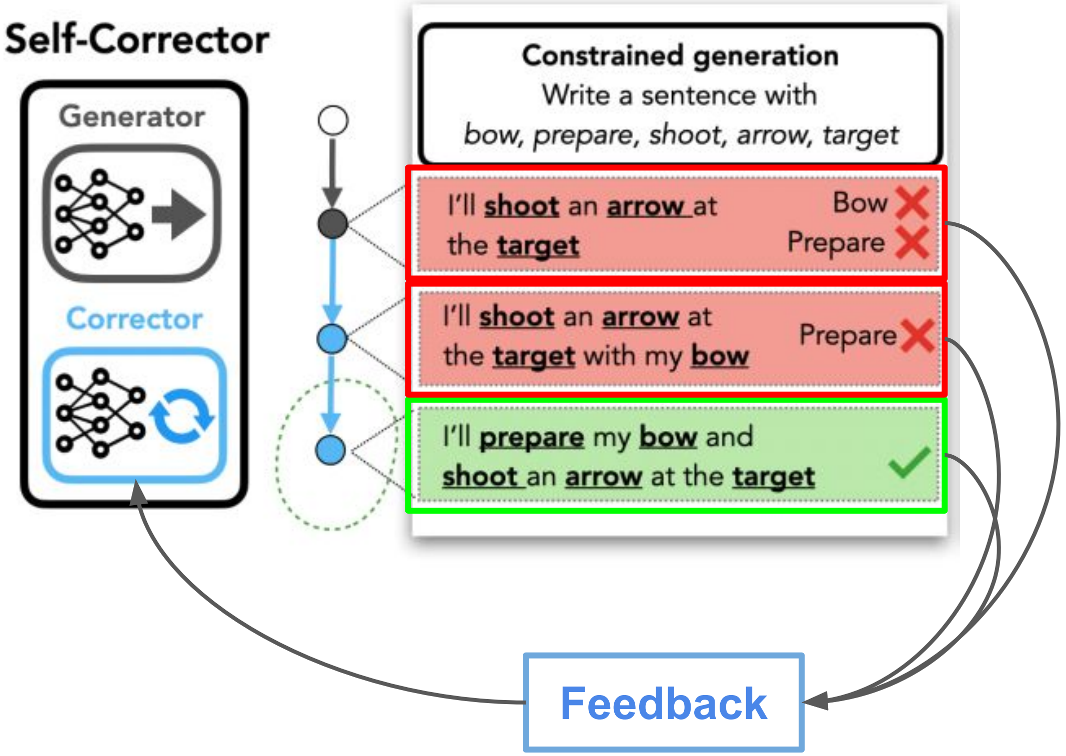
Separate the correction from the generation

Generator - A general-purpose LLM

a) Generate an Initial Hypothesis
b) No Updation

Corrector - Task-Specific Smaller LM

a) Improve on the Initial Hypothesis
b) Updated using feedback



Intuition

Generator

$$p_0(y|x)$$

Corrector

$$p(y|x) = \sum_{y_0} \underbrace{p_0(y_0|x)}_{\text{generator}} \underbrace{p_\theta(y|y_0, x)}_{\text{corrector}}$$

Corrector -> applied multiple times

$$p(y_T|x) = \sum_{y_0} \sum_{y_1} \cdots \sum_{y_{T-1}} p_0(y_0|x) \prod_t p_\theta(y_{t+1}|y_t, x)$$

Method - Learning the Corrector

1) Exploration
2) Pairing
3) Learning
4) Re-Exploration

Exploration

1. Generate Multiple Outputs ($y^{1:N}$) for each Input (x) with decoding scheme (q) (e.g temperature sampling)

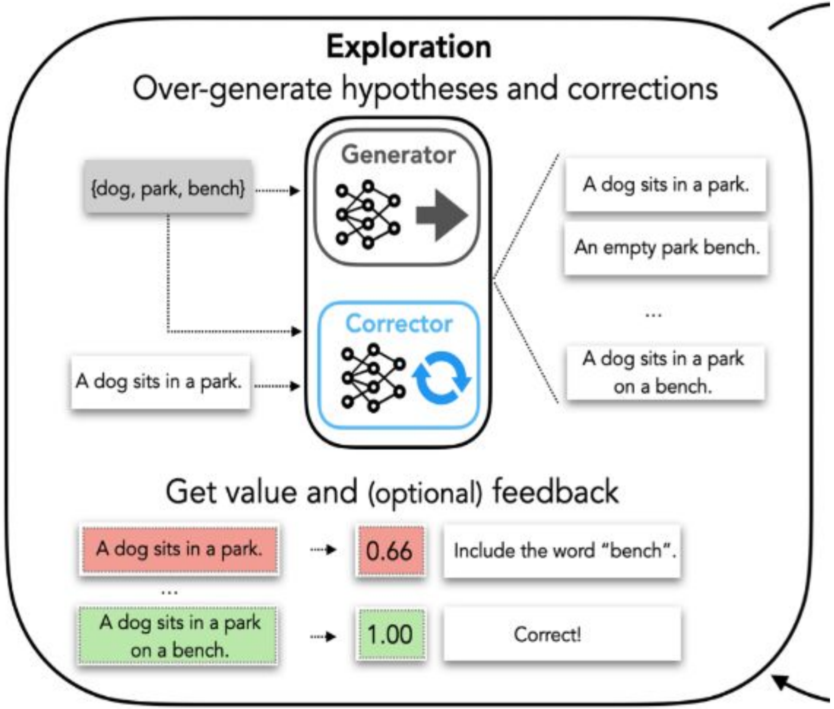
$$D_x = \{(x, y, v(y), f(y)) \mid \text{for all } y \in y^{1:N} \sim q(p_0(\cdot|x))\}$$

2. Get feedback for each y using a defined scalar value function or explicit feedback

Feedback

Scalar Value Function $v : \mathcal{Y} \rightarrow \mathbb{R}$

Explicit Feedback $f : \mathcal{Y} \rightarrow \mathcal{F}$

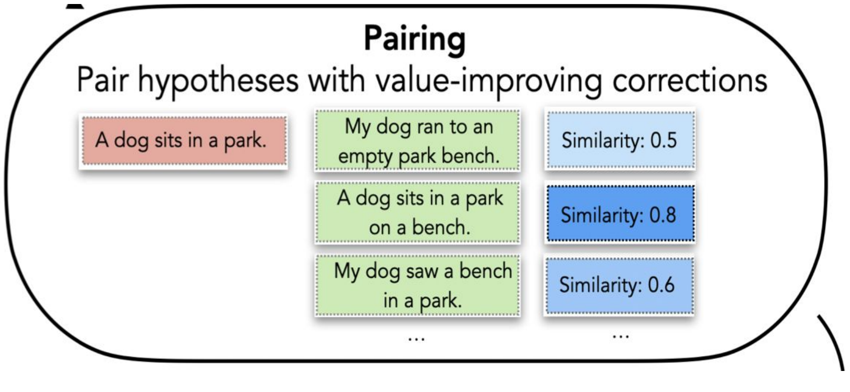


$$x \sim \mathcal{U}(X) \qquad D = \bigcup_{x \in X} D_x$$

Pairing

Form *value-improving pairs* -

$$P_x = \{(x, y, y') \mid v(y) < v(y') \text{ for all } y, y' \in D_x \times D_x\}$$



A pair is formed when an output has a higher value than another.

Learn from "good pairs" - similar pairs with largest absolute difference in values (re: next slide)

$$P = \bigcup_{x \in X} P_x$$

Learning

1. Sample an input x, sample a “good pairs”

$$\mathbb{P}[(x,y,y')|x] \propto \exp\Big(\boxed{\underbrace{\alpha\cdot(v(y')-v(y))}_{\text{improvement}}} + \boxed{\underbrace{\beta\cdot s(y,y')}_{\text{proximity}}}\Big)/\boxed{Z(y)};$$

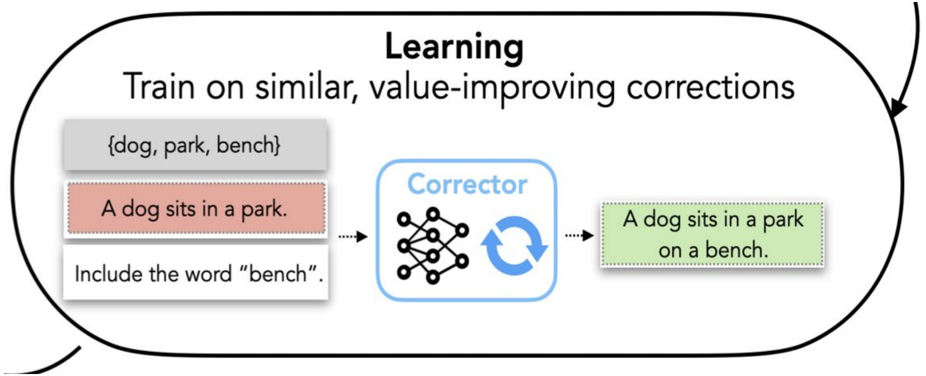
Value-Improving

Similar

Normalization over all available corrections for y

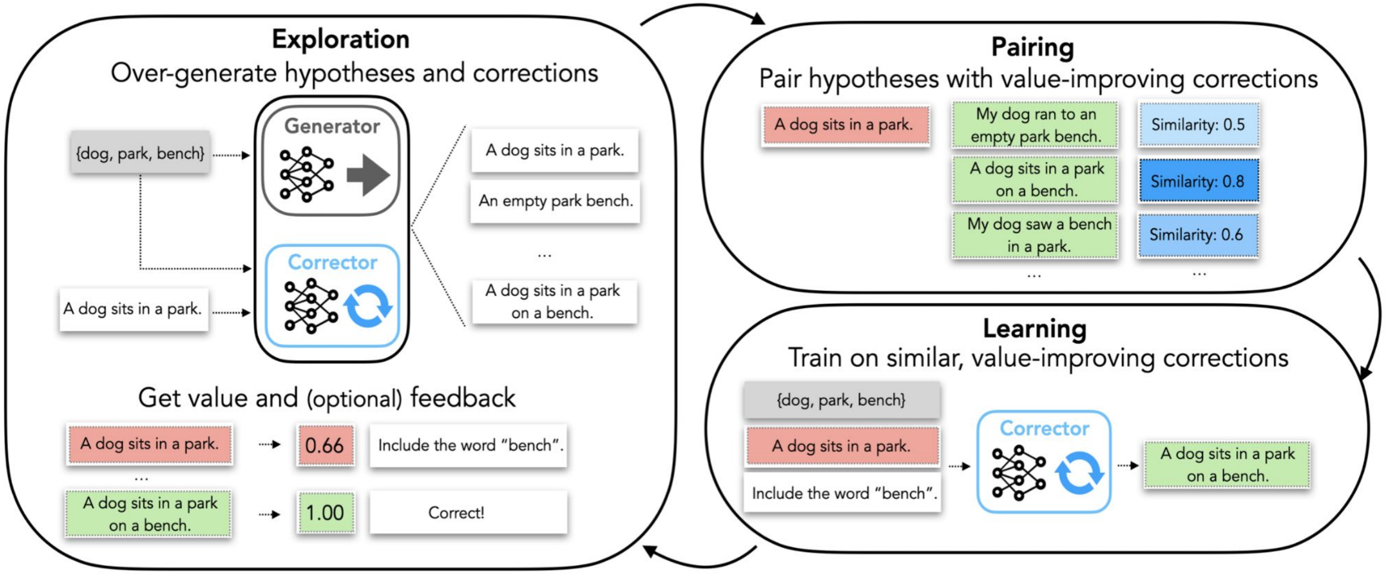
2. Update Corrector - Cross Entropy Loss

$$\mathcal{L}(\theta) = -\log p_\theta(y'|y,x,f(y))$$



Exploration (again)

Add new generations from the corrector into the dataset and re-do the process



Algorithm - Recap

Algorithm 1 Self-corrective learning

input Generator p_0, corrector p_θ, prompts X, value $v(\cdot)$, feedback $f(\cdot)$
  Initialize datapool D by sampling from p_0 ▷ Initialization: Eq. 2
  for iteration $\in \{1, 2, \ldots\}$ **do**
    Form value-improving pairs P from D ▷ Pairing: Eq. 3
    for step in $1, 2, \ldots, M$ **do**
      Sample a batch of value-improving pairs from P using Eq. 4
      Compute the loss and update θ using gradient descent ▷ Learning
    for $x \in X$ **do**
      Sample hypotheses y from datapool D
      Generate corrections $y' \sim p_\theta(\cdot|y, x, f(y))$
      Add all $(x, y', v(y'), f(y'))$ to the datapool D ▷ Exploration: Eq. 5

Inference

1) Decode an initial hypothesis from generator
2) Decode repeatedly from the corrector
    a) Till k
    b) Till a certain objective is reached

Evaluation

1) Improve Generations
2) Correcting Large Generators
3) Leveraging Explicit Natural Language Feedback

3 tasks

a) Low performing task : Program Synthesis
b) Partially Performing Task : Lexical Constrained Generation
c) Open-ended Task : Toxicity Reduction

Using Correctors to improve upon generators

Task 1: Program Synthesis: Given a natural language problem specification x, the task is to generate a program y that upon execution returns the correct answer to x.

1. Generator - GPT-Neo 1.3B (SFT)
   Corrector - GPT-Neo 1.3B
2. Value Function - Binary, No Explicit Feedback
3. Datasets - Multitask, MultiArith, GSM
4. Inference - Greedy Decoding, k = 1

Outperforms Comparable Sized Models (2.7 B)

Dataset	Model	Correct
Multiarith	GPT-NEO 1.3B	60.00
	+SELF-CORRECT	**98.33**
	+SELF-CORRECT$_*$	**99.17**
Multitask	GPT-NEO 1.3B	49.02
	+SELF-CORRECT	**73.53**
	+SELF-CORRECT$_*$	**78.24**

Dataset	Model	Params	Correct
GSM	*OpenAI 3B* [6]	3B	15.50
	OpenAI 6B [6]	6B	20.00
	GPT-NEO [34]	2.7B	18.80
	NEO FCP+PCP [34]	2.7B	19.50
	GPT-NEO	1.3B	8.57
	+SELF-CORRECT	1.3B	**21.26**
	+SELF-CORRECT$_*$	1.3B	**24.22**

*Only on incorrect outputs

Using Correctors to improve upon generators

Problem:
Mrs. Wilsborough saved $500 to buy concert tickets for her family. She bought 2 VIP tickets at $100 each and 3 regular tickets at $50 each. How much of her savings does Mrs. Wilsborough have after she buys the tickets?

Generator:

```
a=2*100
b=3*50
c=a+b
answer=c
print(answer)
```

Corrector:

```
a=2*100
b=3*50
c=500-a-b #fix
answer=c
print(answer)
```

Logical Fix

Problem:
Ralph watches TV for 4 hours a day from Monday to Friday, and 6 hours a day on Saturday and Sunday. How many hours does Ralph spend watching TV in one week?

Generator:

```
a=4*7
b=6*7
c=a+b
answer=c
print(answer)
```

Corrector:

```
a=4*5
b=6*2
c=a+b
answer=c
print(answer)
```

Fixes an incorrect use

Problem:
The pirates plan to explore 4 islands. Two islands require walking 20 miles per day while the other two islands require 25 miles per day. How many miles will they have to walk if it takes 1.5 days to explore each island?

Generator:

```
a=20*2
b=25*2
c=a+b
d=c*1.5
e=d+b
answer=e
print(answer)
```

Corrector:

```
a=20*2
b=25*2
c=a+b
d=c*1.5
answer=d
print(answer)
```

Removes an Incorrect Line

Using Correctors to improve upon generators

Task 2: Lexically Constrained Generation: Given a set of constraint words x, the task is to generate a sentence y that includes all the given constraints.

1. Generator - GPT2 (SFT) , Corrector - GPT2
2. Value Function - Coverage (% of constraints followed)
3. Metrics - Coverage, Fluency (Human Evaluation)
4. Datasets - COMMONGEN (Common Sense Reasoning), E2E
5. Inference - Beam Search, k = 3 with early stopping

Improves Coverage
Maintains Fluency

Method	**Runtime**	**CIDER**	**Constraints**
NeuroLogic [28]	2.04s	14.70	97.70
NeuroLogic-A* [30]	19.24s	15.20	97.80
GPT-2	0.20s	14.97	91.38
SELF-CORRECT	0.80s	15.30	94.58
+NeuroLogic	2.24s	15.28	**97.80**

Method	**Fluency**	**Constraints**
Prefix-Tuning [21]	2.96	91.16
NeuroLogic [28]	2.80	96.91
NeuroLogic-A* [30]	2.85	96.97
GPT-2	2.94	91.50
SELF-CORRECT	**2.98**	**98.77**

Using Correctors to improve upon generators

Constraints:
name : The Mill | Type : restaurant | food : English | price : high | customer rating : average | area : riverside | family friendly : yes | near : Cafe Rouge
Generator:
`The Mill is an expensive, three star restaurant located near Cafe Rouge.`
Corrector:
`The Mill is a children friendly English restaurant in the riverside area near Cafe Rouge. It has an average customer rating and a high price range.`

Constraints:
name : Blue Spice | Type : restaurant | food : Chinese | area : riverside | family friendly : yes | near : Rainbow Vegetarian Cafe
Generator:
`Blue Spice is a restaurant providing Chinese food. It is located in the riverside. It is near Rainbow Vegetarian Cafe.`
Corrector:
`Blue Spice is a family friendly Chinese restaurant in the riverside area near Rainbow Vegetarian Cafe.`

Using Correctors to improve upon generators

Task 3: Toxicity Reduction: Given a prompt x, the task is to generate a fluent continuation y while avoiding offensive content.

1. Generator - GPT2-Large (SFT), Corrector - GPT2-Large
2. Value Function - Perspective API measure Toxicity $v(y) \in [0, 1]$
3. Metrics - Perspective API, Fluency, Diversity
4. Datasets - RealToxicityPrompts
5. Inference - Nucleus Sampling, p = 0.9

Reduces Toxicity

	Toxicity		Fluency	Diversity	
	Avg. Max.	**Prob.**	**Perplexity**	**dist-2**	**dist-3**
GPT-2	0.527	0.520	11.31	0.85	0.85
PPLM [7]	0.520	0.518	32.58	0.86	0.86
GeDi [17]	0.363	0.217	43.44	0.84	0.83
DExpert [27]	0.314	0.128	25.21	0.84	0.84
DAPT [15]	0.428	0.360	31.22	0.84	0.84
PPO [29]	0.218	0.044	14.27	0.79	0.82
Quark [29]	0.196	0.035	12.47	0.80	0.84
SELF-CORRECT	**0.171**	**0.026**	**11.81**	0.80	0.83

Correcting Large Generators

Previous Experiments - Comparable Size of Generator and Corrector

1) Small Generator at Training, Large Generator at Testing
2) Large Generator at Training, Large Generator at Testing

Task	Dataset	Generator (train)	Generator (test)	Generator	Self-corrector
Math Synthesis ↑	GSM	Neo 1.3B	GPT-3	6.96	24.30
		Neo 1.3B	GPT-3 Instruct	36.80	45.00
		GPT-3 Instruct	GPT-3 Instruct	36.80	45.92
Detoxification ↓	RTPrompts	GPT2-L	GPT2-XL	0.383	0.027
		GPT2-L	GPT-3	0.182	0.025
		GPT2-L	GPT-3 Instruct	0.275	0.023

Table 4: **Modularity (program synthesis and detoxification).** Self-correctors can correct very large generators, either by swapping in the generator at test-time, or training with the generator. For math synthesis, the corrector is GPT-Neo 1.3B, and here we only correct incorrect outputs. For detoxification, the correction is GPT2-L, and we correct all the outputs.

Leveraging Explicit Feedback

Use explicit feedback as the natural language feedback

Claim: Correctors learn to use the feedback.

Leveraging Explicit Feedback

Program Synthesis: Prompt a LLM to get feedback

1) Problem
2) Hypothesis
3) Gold Solution
4) Demonstrations of feedback -

   In the initial guess, 3 should be subtracted

Leveraging Explicit Feedback

Program Synthesis: Prompt a LLM to get feedback

1) Problem
2) Hypothesis
3) Gold Solution
4) Demonstrations of feedback - *In the initial guess, 3 should be subtracted*

Also done at inference: Possible Leakage?

Leveraging Explicit Feedback

Lexical Constraints: Mention the lexical constraint in natural language

Constraints: dog, park, bench

Hypothesis: “There is a dog in the park”

Explicit Feedback: “adding constraint word: bench”

Correction: “The dog is sitting near the bench in the park”

Leveraging Explicit Feedback

Toxicity: Perspective API provides fine-grained feedback on toxicity score.

Profanity Score: 0.8

Identify Attack Score: 0.9

Training Feedback: Largest change in corrected attribute b/w correction and hypothesis as Natural Language

Inference Feedback: Pick the attribute with the largest score

Multiple Corrections

Multiple Corrections ~ Better Performance

Performance Plateau soon after

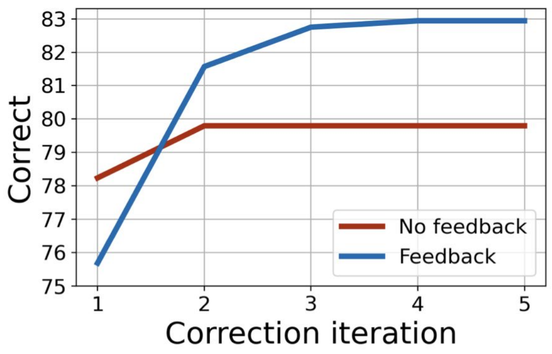


Figure 5: Math: multiple corrections.

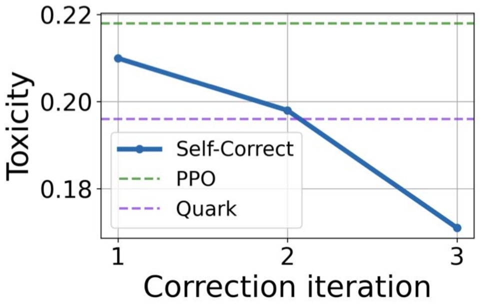


Figure 4: Applying multiple corrections reduces toxicity.

Feature Ablation

$$\mathbb{P}[(x, y, y')|x] \propto \exp\Big(\boxed{\underbrace{\alpha \cdot (v(y') - v(y))}_{\text{improvement}}} + \boxed{\underbrace{\beta \cdot s(y, y')}_{\text{proximity}}}\Big)/Z(y),$$

Value-Improving

Similar

Ablation	Math	CommonGen
Self-Correct	**78.24**	**94.55**
✗ proportional sampling	77.25	93.49
✗ value pairing	62.35	91.76

Does Exploration actually help?

1) Exploration only with Base Generator
2) Exploration with Corrector Generator

Exploration	**Multiarith**	**Multitask**	**GSM8k**
✗	89.20	73.49	17.60
✓	**99.17**	**78.24**	**23.96**

Strengths

1) Efficient & Smaller Task-Specific LM
2) Assume API-access to LLM
3) Continuous refinement

Weaknesses

Some choices in evaluation

1) Possible Leakage in Lexical Evaluation
2) Difference in Inference strategies between all tasks

Task	Dataset	Generator (train)	Generator (test)	Generator	Self-corrector
Math Synthesis ↑	GSM	Neo 1.3B	GPT-3	6.96	24.30
		Neo 1.3B	GPT-3 Instruct	36.80	45.00
		GPT-3 Instruct	GPT-3 Instruct	36.80	45.92
Detoxification ↓	RTPrompts	GPT2-L	GPT2-XL	0.383	0.027
		GPT2-L	GPT-3	0.182	0.025
		GPT2-L	GPT-3 Instruct	0.275	0.023

Table 4: **Modularity (program synthesis and detoxification).** Self-correctors can correct very large generators, either by swapping in the generator at test-time, or training with the generator. For math synthesis, the corrector is GPT-Neo 1.3B, and here we only correct incorrect outputs. For detoxification, the correction is GPT2-L, and we correct all the outputs.

Weaknesses/Follow-up work

1) More examples (especially on explicit feedback)
2) Unexplored Settings -
    a) Training on Large Generator, Testing using Small Generator
    b) Large Generator Evaluation on Lexical Constraints
    c) Exploration help with other tasks?

2) Ambiguous Tasks: Not every task has a value function/automated feedback

3) Self-Correct? :)

Thanks!