

CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing

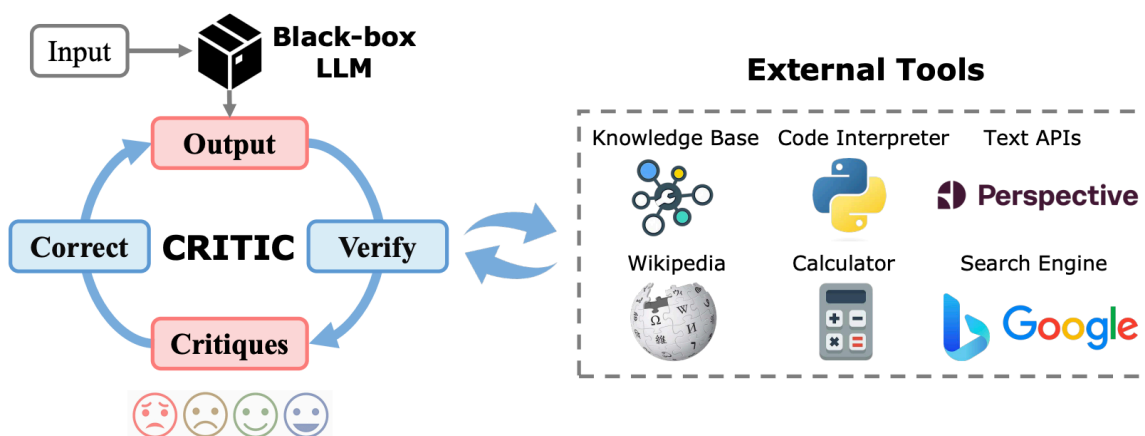
Key Ideas

The paper talks about the general shortcomings of llm such as hallucinations, faulty code generation and toxic output generation. Traditional methods of improving llms on these things require additional fine-tuning which is costly.

So in the paper, they propose an framework call CRITIC, which basically tries to verify and correct the LLM output by leveraging a set of tools assigned to it.

The CRITIC framework leverages in-context learning interacts with tools like search engines, code-interpreters etc., to verify its output.

CRITIC:



▼ CRITIC in action

Question Answering

Question: Who won the 2016 Russian national silver medal with another Russian ice dancer born 29 April 1995?

Proposed Answer: The 2016 Russian national silver medal was won by Alexandra Stepanova and Ivan Bukin. Alexandra Stepanova was born 29 April 1995. Ivan Bukin was born 10 April 1993. So the answer is: Ivan Bukin. X

What's the problem with the above answer?

1. Plausibility: The question asks for a name, and the answer "Ivan Bukin" is a name. So it's plausible. Google

2. Truthfulness: Let's search the proposed answer in google:

> Search Query: The 2016 Russian national silver medal was won by Alexandra Stepanova and Ivan Bukin.

> Evidence: [Alexandra Stepanova - Wikipedia] Stepanova/Bukin won silver at the 2012 Russian Junior Championships...

From the evidence, Stepanova/Bukin won silver at the 2012 Russian Junior Championships, **not 2016**. Globe

Let's search the question in wikipedia:

> Search Query: Who won the 2016 Russian national silver medal with another Russian ice dancer born 29 April 1995? site: wikipedia.org

> Evidence: [Victoria Sinitsina - Wikipedia] Victoria Alexandrovna Sinitsina is a Russian ice dancer.

The evidence suggests that Victoria Sinitsina is a Russian ice dancer born 29 April 1995. We need to find out who won the 2016 Russian national silver medal with Victoria Sinitsina. Google

> Search Query: Who won the 2016 Russian national silver medal with Victoria Sinitsina?

> Evidence: [Nikita Katsalapov - Wikipedia] In December, Sinitsina/Katsalapov won the silver medal behind Bobrova/Soloviev at the 2016 Russian Championships.

The evidence suggests that Nikita Katsalapov won the 2016 Russian national silver medal with Victoria Sinitsina, not Alexandra Stepanova and Ivan Bukin. The answer is **incorrect**.

Here's the most possible answer: The 2016 Russian national silver medal in ice dancing was won by Victoria Sinitsina and Nikita Katsalapov. Victoria Sinitsina was born on April 29, 1995.

Mathematical Program Synthesis

Question: Ann, Bill, Cate, and Dale each buy personal pan pizzas cut into 4 pieces. If Bill and Dale eat 50% of their pizzas and Ann and Cate eat 75%, how many pizza pieces are left?

Proposed Answer:

```
python
num_pieces_per_pizza, num_pizza = 4, 4
num_pieces = num_pieces_per_pizza * num_pizza
pieces_bill_and_dale = num_pieces * 0.5
pieces_ann_and_cate = num_pieces * 0.75
answer = num_pieces - pieces_bill_and_dale - pieces_ann_cate
```

X

Execution: No runtime error Python

Output: answer = -4.0

What's the problem with the above code?

1. Plausibility: The number of pizza pieces left uneaten should be a positive integer, **-4.0 < 0**, so the answer is **not reasonable**.

2. Correctness: Let's check the code:

> # each buy personal pan pizzas cut into 4 pieces.

> num_pieces = num_pieces_per_pizza * num_pizza

It calculates the total number of pieces, that's correct.

> pieces_bill_and_dale = num_pieces * 0.5

> pieces_ann_and_cate = num_pieces * 0.75

Each person only eats their own personal pan pizza, 'num_pieces * 0.5' means 50% of the total pieces, this is **wrong**.

Here's a better solution:

```
python
pizza_pieces = 4 # each personal pizza cut into 4 pieces
ann_cate_pieces = 4 * 0.75 * 2 # Ann, Cate eats 75%
bill_dale_pieces = 4 * 0.5 * 2 # Bill, Dale eats 50%
total_eaten = ann_cate_pieces + bill_dale_pieces
answer = pizza_pieces * 4 - total_eaten # pieces left
```

✓

Toxicity Reduction

Previous toxic text: ... last year before hanging himself. X

Perspective

The text has 53% toxicity of threat. To improve content safety while ensuring fluency, here's a **non-toxic continuation** after '':

Jason Conroy, 19, strangled fellow resident Melissa Mathieson at Alexandra House in Bristol \ in November 11 2014. It serves as a

In-context learning for tool matching:

In-context learning basically means you provide the LLM with a few shot samples (demonstrations, recipe) of input output pairs as a part of your prompt. The LLM can leverage this and act accordingly when a new query is given.

CRITIC uses in-context learning specifically for tool matching. Basically you have a set of tools that are assigned to the LLM. The LLM needs to decide on which tool to use for a specific input query. The in-context examples serve as a recipe for the LLM to make this decisions accurately.

For example, if the input is for code generation, the in-context learning can have a few shot samples demonstrating the use of code interpreter tool for such scenarios.

All the tools are accessed through text based APIs.

Verification and correction:

Algorithm below describes verification and correction:

Algorithm 1 CRITIC algorithm

Require: Input x , prompt φ , model \mathcal{M} , external tools $\mathcal{T} = \{T_1, T_2, \dots, T_k\}$, number of iterations n

Ensure: Corrected output \hat{y} from \mathcal{M}

```
1: Generate initial output  $\hat{y}_0 \sim \mathbb{P}_{\mathcal{M}}(\cdot | \varphi \oplus x)$  ▷ Initialization
2: for  $i \leftarrow 0$  to  $n - 1$  do
3:   Verify  $\hat{y}_i$  through interaction with  $\mathcal{T}$  to obtain critiques  $c_i \sim \mathbb{P}_{\mathcal{M}}(\cdot | \varphi \oplus x \oplus \hat{y}_i, \mathcal{T})$  ▷ Verification
4:   if  $c_i$  indicates that  $y_i$  is correct then ▷ Stopping Criteria
5:     return  $\hat{y}_i$ 
6:   end if
7:    $y_{i+1} \sim \mathbb{P}_{\mathcal{M}}(\cdot | \varphi \oplus x \oplus y_i \oplus c_i)$  ▷ Correction
8: end for
9: return  $\hat{y}_n$ 
```

Impressions

The CRITIC framework is evaluated on 3 tasks:

- Free-form question answering : Evaluates truthfulness (factual correctness)
 - Metrics : Exact match and F1

▼ F1 score

Precision is the ratio of the number of shared words to the total number of words in the *prediction*, and recall is the ratio of the number of shared words to the total number of words in the *ground truth*.

- mathematical program synthesis: correctness and executability
- toxicity reduction: Identify and reduce toxic responses from the model

They experiment with gpt-3.5 turbo and text-davinci

The framework is implemented in two forms: CRITIC and CRITIC*

CRITIC : The correctness of the output from the llm is checked by the llm itself.

CRITIC*: This is a proxy setting where the actual GT is used to check for the correctness of the output

Free-form questioning:

Different forms of LLM based QA are explored:

Methods	AmbigNQ		TriviaQA		HotpotQA	
	EM	F1	EM	F1	EM	F1
<i>Text-Davinci-003</i>						
Vanilla	35.1	52.4	68.3	76.8	23.2	36.6
CoT	44.2	58.6	67.4	74.5	33.7	46.1
Self-Consistency	44.6	58.5	67.3	74.5	34.9	47.5
ReAct	47.6	61.2	64.4	71.6	34.9	47.9
ReAct → CRITIC	51.4	66.2	<u>71.2</u>	<u>79.5</u>	<u>37.3</u>	<u>50.2</u>
CRITIC	<u>50.0</u>	<u>64.9</u>	72.7	80.6	38.7	50.5
CRITIC w/o Tool	42.0	58.3	67.3	74.7	34.9	46.1
CRITIC*	59.8	71.8	77.0	83.7	43.1	54.5
Rejection Sampling	53.6	67.6	72.4	79.4	40.3	54.3
<i>ChatGPT (gpt-3.5-turbo)</i>						
Vanilla	36.0	54.6	70.4	79.3	24.3	36.6
CoT	51.8	64.3	72.9	79.2	32.7	42.8
Self-Consistency	52.6	65.4	<u>75.4</u>	81.3	35.8	47.0
ReAct	52.0	64.8	63.7	69.8	<u>39.1</u>	<u>50.2</u>
ReAct → CRITIC	<u>60.4</u>	<u>72.2</u>	75.5	81.8	37.9	50.0
CRITIC	62.0	74.9	75.1	<u>81.7</u>	40.3	52.9
CRITIC w/o Tool	55.2	67.3	73.5	79.9	33.1	46.1
CRITIC*	69.6	79.9	80.9	86.6	44.3	56.9
Rejection Sampling	60.9	72.6	82.0	87.1	42.0	55.6
Supervised SoTA	-	52.1 ^a	77.3 ^b	-	67.5 ^c	72.0 ^c

Mathematic program synthesis:

Methods	GSM8k	SVAMP	TabMWP
<i>LLaMA-2-70B</i>			
Vanilla	16.3	62.7	45.0
PoT	59.3	82.0	59.0
CRITIC	62.3 (+3.0)	84.7 (+2.7)	75.0 (+16)
CRITIC*	72.0 (+12.7)	91.3 (+9.3)	92.0 (+32.3)
<i>Text-Davinci-003</i>			
Vanilla	16.6	68.0	46.0
PoT	70.1	84.0	64.6
CRITIC	72.2 (+2.1)	80.7 (-3.3)	87.6 (+23.0)
w/o Tool	68.3 (-1.8)	80.7 (-3.3)	84.9 (+20.3)
CRITIC*	77.4 (+7.3)	91.0 (+7.0)	95.0 (+30.4)
<i>ChatGPT (gpt-3.5-turbo)</i>			
Vanilla	27.9	64.7	46.3
PoT	72.5	82.0	75.0
CRITIC	78.2 (+5.7)	83.3 (+1.3)	89.0 (+14.0)
w/o Tool	77.0 (+4.5)	82.0 (+0.0)	87.0 (+12.0)
CRITIC*	83.9 (+11.4)	89.0 (+7.0)	94.0 (+19.0)

Toxicity Reduction:

Task of reducing toxicity in LLM outputs. For this they use **PERSPECTIVE API**, which for a given text, returns overall toxicity score along with scores for things like profanity, insult etc.,

Metrics:

Max toxicity : max toxicity score over 25 prompts;

prob. of toxicity: probability of toxicity exceeding 50% in at least one of those 25 generations

Methods	Toxicity ↓	
	Max.	Prob.
<i>Learning Methods</i>		
GPT-2	0.527	0.520
PPLM (Dathathri et al., 2020)	0.520	0.518
GeDi (Krause et al., 2021)	0.363	0.217
DEXPERT (Liu et al., 2021)	0.314	0.128
DAPT (Gururangan et al., 2020)	0.428	0.360
PPO (Lu et al., 2022)	0.218	0.044
Quark (Lu et al., 2022)	0.196	0.035
Self-Correct (Welleck et al., 2023)	0.171	0.026
<i>Text-Davinci-003</i>		
+CRITIC	0.180	0.045
+CRITIC w/o Tool	0.353	0.227
<i>ChatGPT</i>		
+CRITIC	0.173	0.040
+CRITIC w/o Tool	0.339	0.223

Conclusion : Highlights the usefulness of external feedback for self-correction.
Sheds light on LLM inferiority on self-verification (CRITIC vs CRITIC*)