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

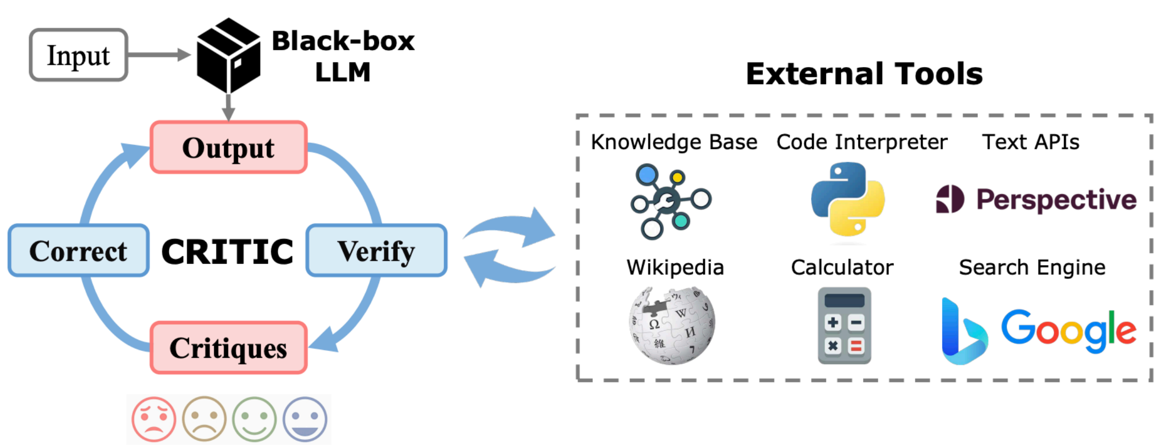
Key Ideas

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.

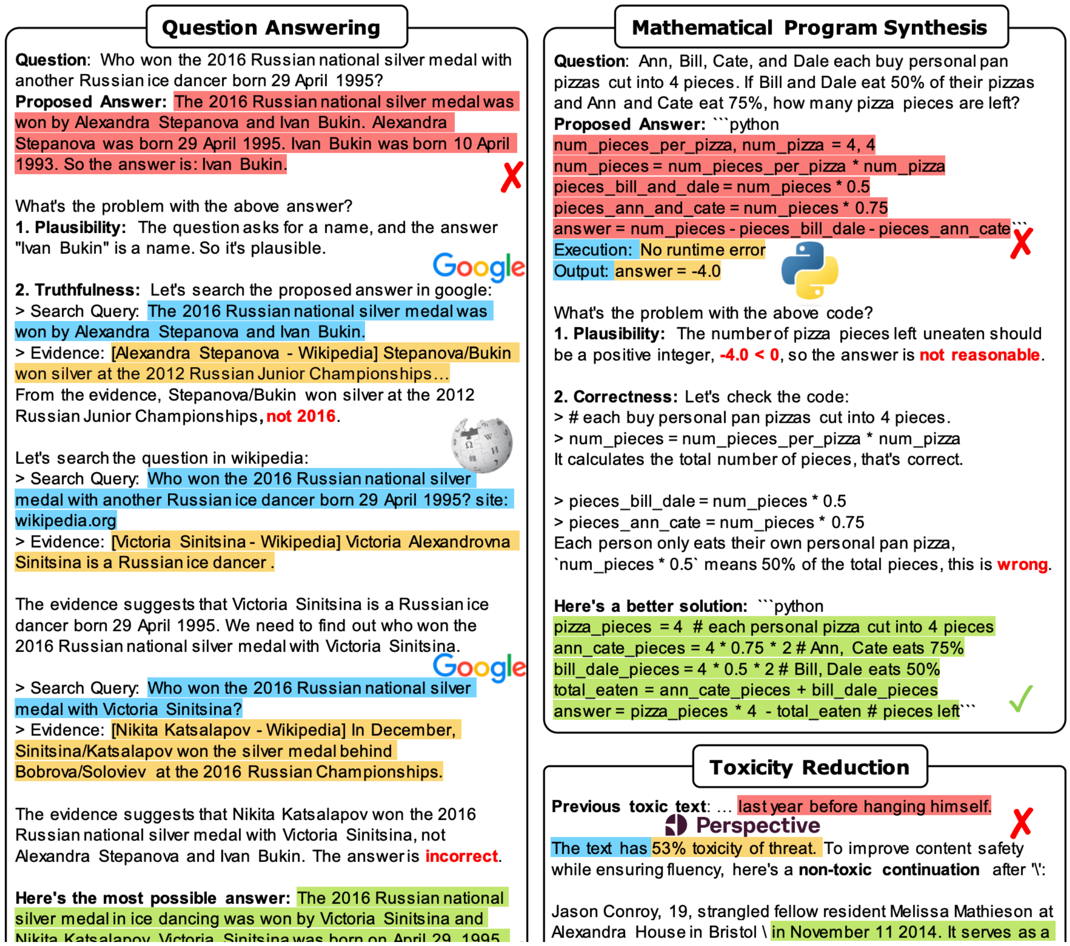
So in the paper, they propose an framework call CRITIC, which basically tries to verify and correct the LLM   output by leveraging a set of tools assigned to it.

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

CRITIC:



▼ CRITIC in action



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 $\wp$, model $\mathcal{M}$, external tools $\mathcal{T} = \{T_1, T_2, ..., T_k\}$, number of iterations n
Ensure: Corrected output $\hat{y}$ from $\mathcal{M}$

Generate initial output $\hat{y_0} \sim \mathbb{P}_{\mathcal{M}}(\cdot|\wp \oplus x)$ ▷ Initialization
for $i \leftarrow 0$ to $n-1$ **do**
  Verify $\hat{y}_i$ through interaction with $\mathcal{T}$ to obtain critiques $c_i \sim \mathbb{P}_{\mathcal{M}}(\cdot|\wp \oplus x \oplus \hat{y}_i, \mathcal{T})$ ▷ Verification
  if c_i indicates that y_i is correct **then** ▷ Stopping Criteria
    return $\hat{y}_i$
  end if
  $\hat{y_{i+1}} \sim \mathbb{P}_{\mathcal{M}}(\cdot|\wp \oplus x \oplus y_i \oplus c_i)$ ▷ Correction
end for
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
			Text-Davinci-003			
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**	71.2	79.5	37.3	50.2
CRITIC	50.0	64.9	**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
			ChatGPT (gpt-3.5-turbo)			
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	75.4	81.3	35.8	47.0
ReAct	52.0	64.8	63.7	69.8	39.1	50.2
ReAct → CRITIC	60.4	72.2	**75.5**	**81.8**	37.9	50.0
CRITIC	**62.0**	**74.9**	75.1	81.7	**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
		LLaMA-2-70B	
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)**
		Text-Davinci-003	
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)**
		ChatGPT (gpt-3.5-turbo)	
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.**
	Learning Methods	
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
Text-Davinci-003	0.344	0.210
+CRITIC	**0.180**	**0.045**
+CRITIC w/o Tool	0.353	0.227
ChatGPT	0.325	0.192
+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*)