



RoCoIns: Enhancing Robustness of Large Language Models through Code-Style Instructions

Fudan NLP



- Large language models (LLMs) have shown increasing power in following human instructions and solving various NLP tasks.
- Recent evaluations in terms of LLMs have revealed their insufficient robustness when prompted with instructions containing textual adversarial samples, raising concerns about their real-world applications.



Table 2: Performance and robustness test results (accuracy) of GPT series models in zero-shot and few-shot scenarios on **SemEval2014-Laptop** dataset.

Model	AddDiff # 331 samples		ReverseNonTarget # 104 samples		ReverseTarget # 331 samples	
	ori	trans	ori	trans	ori	trans
<i>0-shot</i>						
code-davinci-002	92.88±2.14	90.18±7.42	91.39±2.96	53.09±2.78	93.23±1.65	58.61±0.89
text-davinci-001	85.21±1.70	80.10±2.11	85.89±1.68	47.35±4.16	85.26±2.17	53.56±0.92
text-davinci-002	86.38±0.11	81.90±0.35	85.57±0.21	52.97±2.96	86.40±0.26	56.68±5.17
text-davinci-003	83.84±0.33	77.50±2.43	82.43±0.42	39.61±4.83	83.62±0.12	47.04±4.64
gpt-3.5-turbo	85.57±1.27	86.55±8.67	88.78±2.22	41.78±7.36	85.67±1.36	49.75±9.51
<i>1-shot</i>						
code-davinci-002	96.33±0.58	92.67±0.58	94.00±1.00	53.33±1.53	96.33±0.58	65.00±1.00
text-davinci-001	82.87±1.04	72.69±0.63	82.85±1.68	45.74±2.29	82.94±0.88	47.50±3.15
text-davinci-002	86.05±0.43	82.20±1.98	85.22±0.24	55.18±2.38	86.05±0.43	56.77±3.08
text-davinci-003	85.77±0.69	87.17±4.62	85.63±1.05	52.22±8.86	85.84±0.57	57.07±7.43
gpt-3.5-turbo	88.99±0.73	85.17±4.90	93.22±1.00	41.93±5.21	89.18±0.90	47.95±6.93
<i>3-shot</i>						
code-davinci-002	97.00±1.00	94.00±1.00	94.00±0.00	52.00±2.65	97.00±1.00	64.33±1.53
text-davinci-001	83.33±0.69	71.90±0.12	83.40±0.24	48.40±1.98	83.44±0.87	50.26±0.77
text-davinci-002	85.41±0.43	81.55±1.86	84.80±0.97	54.01±2.28	85.48±0.33	56.65±2.43
text-davinci-003	85.91±0.12	88.73±4.11	85.08±0.48	55.59±11.11	85.98±0.12	59.14±7.11
gpt-3.5-turbo	90.75±1.57	90.23±6.40	93.09±1.98	47.93±5.83	90.45±1.08	53.62±4.82

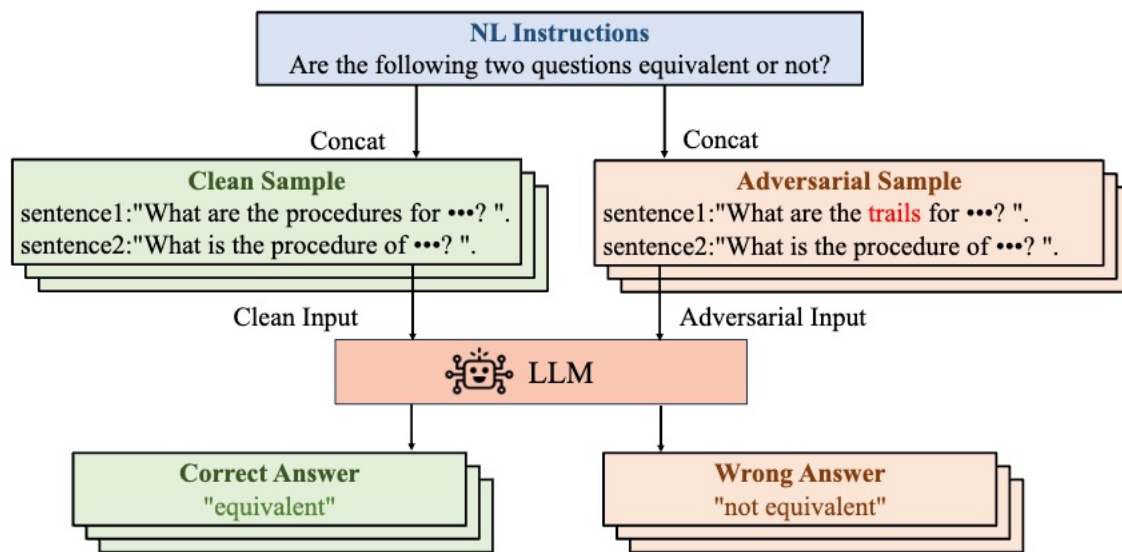


Traditional Methods In response to textual adversarial attacks

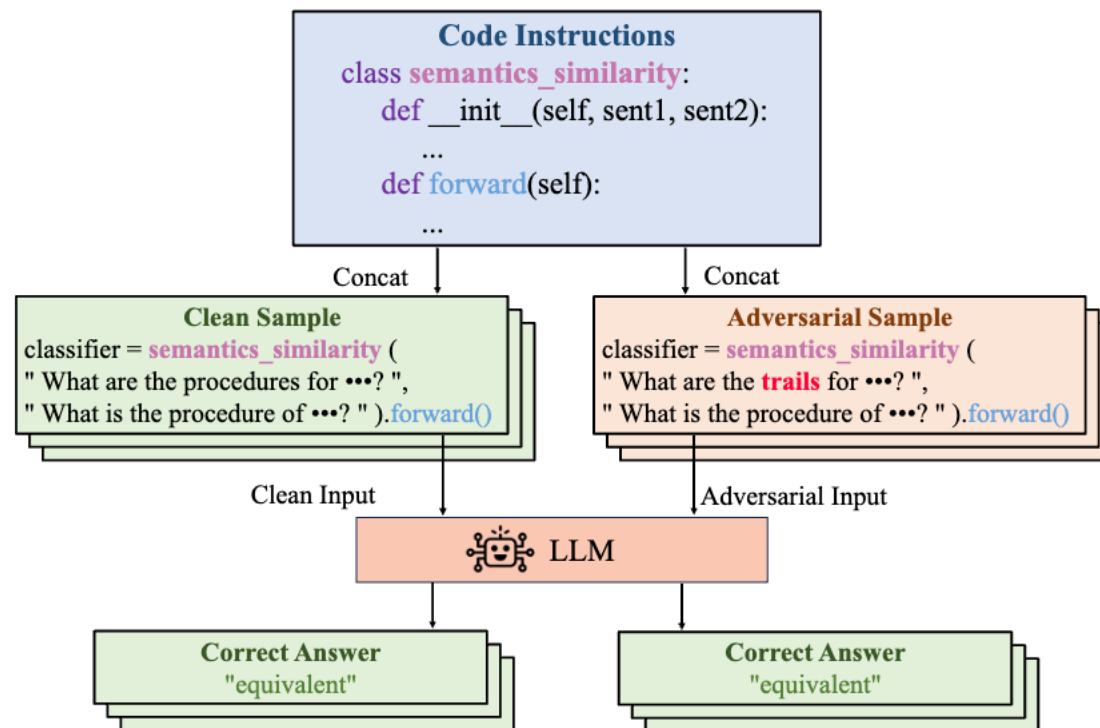
- adversarial training
- interval bound propagatio
- randomized smoothing

DrawBacks beyond LLMs

- parameters update of models
- not access to LLMs



(a) Prompt LLMs with natural language instructions



(b) Prompt LLMs with code-style instructions



Code-Style Instructions

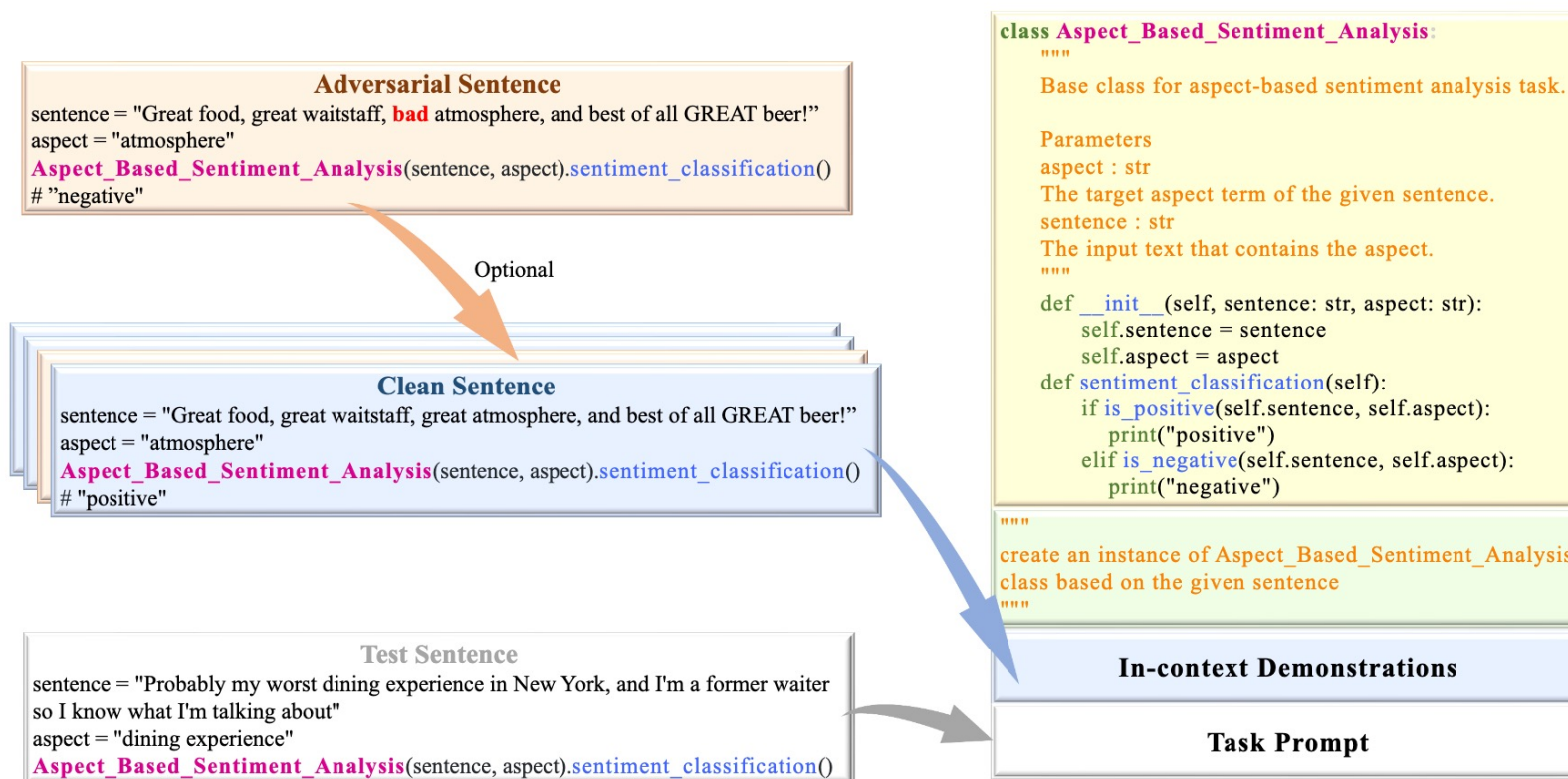
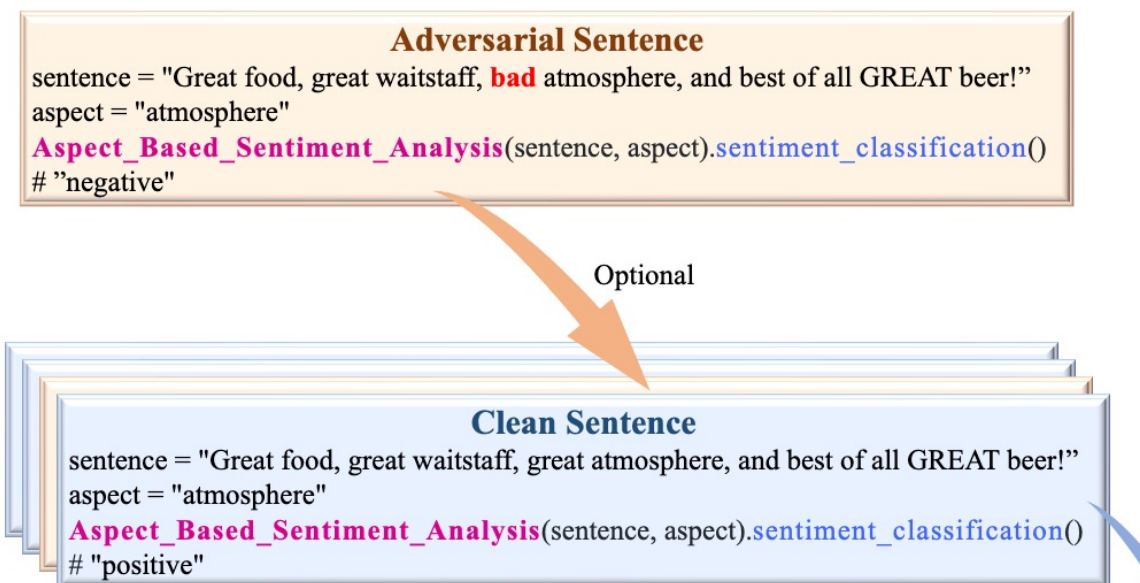


Figure 2: Components of code-style instructions. (1) Class definition mainly contains the class name, annotation, initial function and implementation function. (2) In-context demonstrations consist of k (adversarial) samples in the corresponding code style. (3) Task prompt follows the same format as demonstrations without a ground truth label.



Adversarial Context Methods

- Recent studies have shown that ICL can be regarded as a form of implicit fine-tuning
- Additionally, introducing adversarial samples prompts the LLMs to recognize specific adversarial attacks, such as spelling errors and word substitutions.





Models

- text-davinci-003
- gpt-3.5-turbo

Datasets

- AdvGLUE
- Restaurant

Metrics

- Attack Success Rate (ASR)
- Clean Accuracy

$$ASR = \sum_{(x,y) \in T} \frac{\mathbb{1}[f(A(x)) \neq y]}{\mathbb{1}[f(x) = y]}$$



Model	Method	Dataset(ASR)								Avg(ASR)	Avg(Acc)
		AdvGLUE					Restaurant-T				
		SST-2	QQP	MNLI	QNLI	RTE	RevTgt	RevNon	AddDiff		
Random		50.0	50.0	66.7	50.0	50.0	66.7	66.7	66.7	58.35	41.67
Zero-Shot											
davinci-003	NL	44.6	55.1	44.6	38.5	34.6	44.11	20.00	13.19	36.84	—
gpt-3.5-turbo	NL	39.9	18.0	32.2	34.5	24.74	49.42	36.09	42.67	34.68	—
Few-Shot											
Shot Number		4	6	6	4	4	6	6	6		
davinci-003	NL	25.39	23.94	25	24.03	15.18	25.09	11.39	10.16	20.02	72.07
	CoT	23.07	26.56	23.8	23.62	14.28	24.04	9.35	7.65	19.05	73.43
	Code	23.97	25.35	22.54	21.73	12.65	23.52	5.79	5.08	17.58	75.82
	Code+adv	20.93	22.53	22.33	20.21	12.65	21.97	6.52	4.66	16.47	77.20
gpt-3.5-turbo	NL	19.23	23.07	21.73	18.75	22.05	29.19	17.03	16.4	20.93	70.45
	CoT	21.08	20.63	14.85	18.34	21.42	34.67	14.02	14.28	19.91	71.14
	Code	17.83	18.46	18.55	14.28	21.43	23.35	14.4	10.08	17.29	74.73
	Code+adv	16.43	9.23	14.4	10.71	22.85	22.43	13.4	12.71	15.27	76.13



Perplexity: Code vs NL

Utilizing the pre-trained code model with code-style instructions consistently results in the lowest perplexity

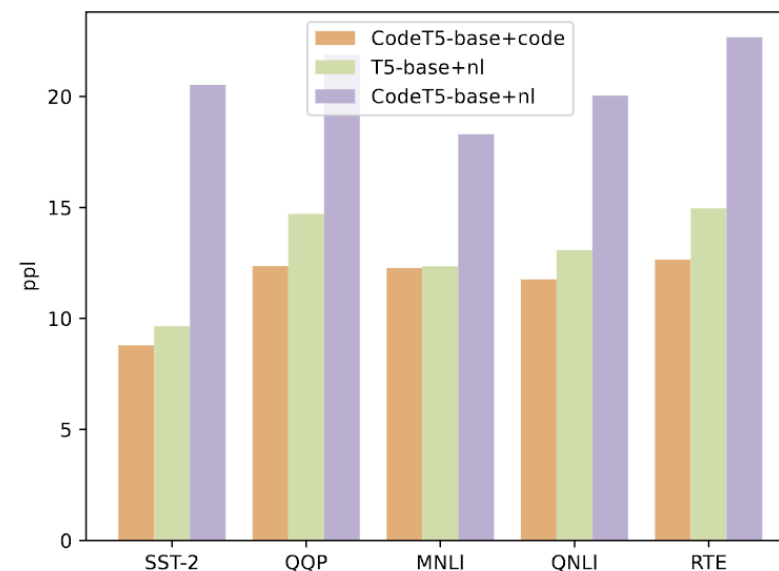


Figure 3: Perplexity for AdvGLUE dataset on T5-base with natural language instructions and CodeT5-base with both natural language and code-style instructions. We report the logarithm of their initial values.



Different Code-style Instructions

Code-style instructions almost always outperform natural language instructions with lower ASR, showcasing the overall superiority of code-style instructions

Prompt	SST-2	MNLI	QNLI
NL	19.23	21.73	18.75
NL(complicated)	19.45	21.64	18.97
Class Exec	17.83	18.55	14.28
Class Init	20.93	16.12	16.52
Func Exec	18.6	17.52	14.03

Table 3: ASR for different code-style instructions design. "class exec" is the code format used in our main experiments and is highlighted in light grey. The experiment is conducted on gpt-3.5-turbo.



Number of In-context Demonstrations

Completeness and balance of labels are significant for the task performance.

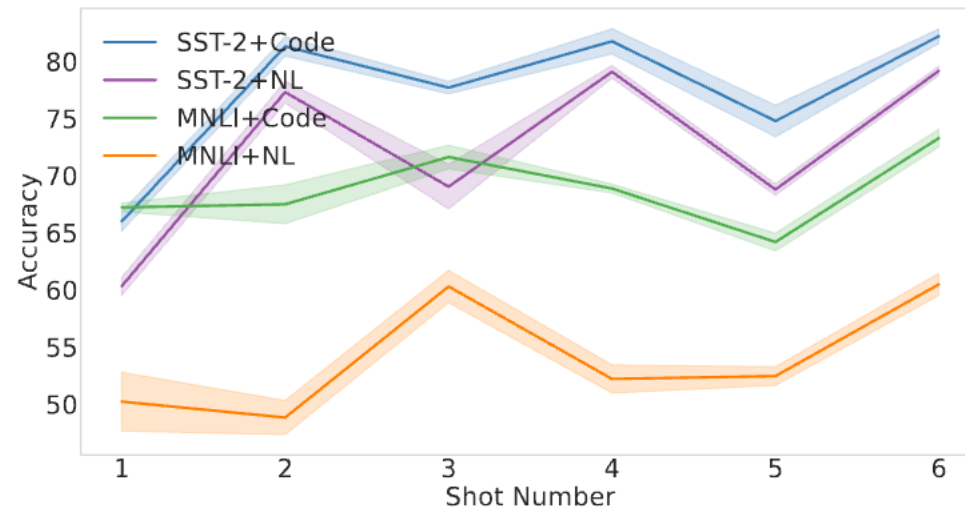


Figure 4: Accuracy with the different number of in-context demonstrations on SST-2 and MNLI adversarial dataset. The experiment is conducted on gpt-3.5-turbo.



Visualization Analysis

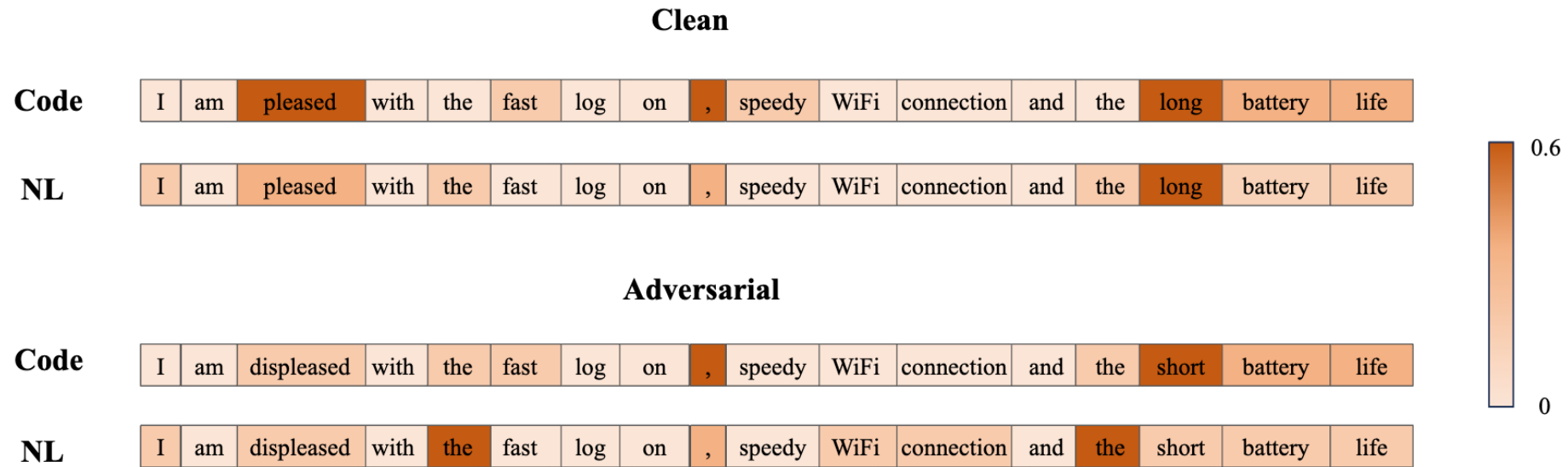


Figure 5: Visualization of a sample's gradient on each word when fine-tuning CodeT5 with code-style instruction and T5 with natural language instruction, respectively. The sample is selected from the Restaurant-T dataset with both its clean and adversarial versions. The sample aims to determine the sentiment polarity of the aspect "*battery life*" in the sentence with "*positive*" or "*negative*".



In this paper, we propose RoCoIns to utilize code-style instructions instead of natural language instructions to enhance the robustness of closed source black-box models against textual adversarial attacks.

Instructions in code style, which are more structural and less ambiguous than natural language instructions, provide LLMs with more precise instructions.

Besides, we propose adversarial context method to further boost the robustness. Experiments show that our method consistently outperforms prompting LLMs with natural language instructions under the few-shot setting. We conduct further analysis to verify the advantages of using code-style instructions.