An Empirical Study to Evaluate AIGC Detectors on Code Content

Jian Wang[1,2], Shangqing Liu[1], Xiaofei Xie[2], **Yi Li**[1]

1. Nanyang Technological University
2. Singapore Management University

ASE'24 – Sacramento, California
Oct 29, 2024

"AND I PROMISE THESE TERM PAPERS WON'T BE DETECTED AS AI" stack overflow

***All* use of generative AI (e.g., ChatGPT[1] and other LLMs) is banned when posting content on Stack Overflow.**

This includes "asking" the question to an AI generator then copy-pasting its output *as well as* using an AI generator to "reword" your answers.

Please see the Help Center article: What is this site's policy on content generated by generative artificial intelligence tools?

Overall, because the average rate of getting *correct* answers from ChatGPT and other generative AI technologies is too low, **the posting of content created by ChatGPT and other generative AI technologies is *substantially harmful* to the site and to users who are asking questions and looking for *correct* answers.**

ChatGPT

Source: https://stackoverflow.com/help/gen-ai-policy

scikit-learn is one of the sample projects featured in SWE-bench
https://openai.com/index/introducing-swe-bench-verified/

adrinjalali commented on Jul 10

@DocInspector 's bio on GH says:

> We prefer to remain anonymous for double-blind paper review.

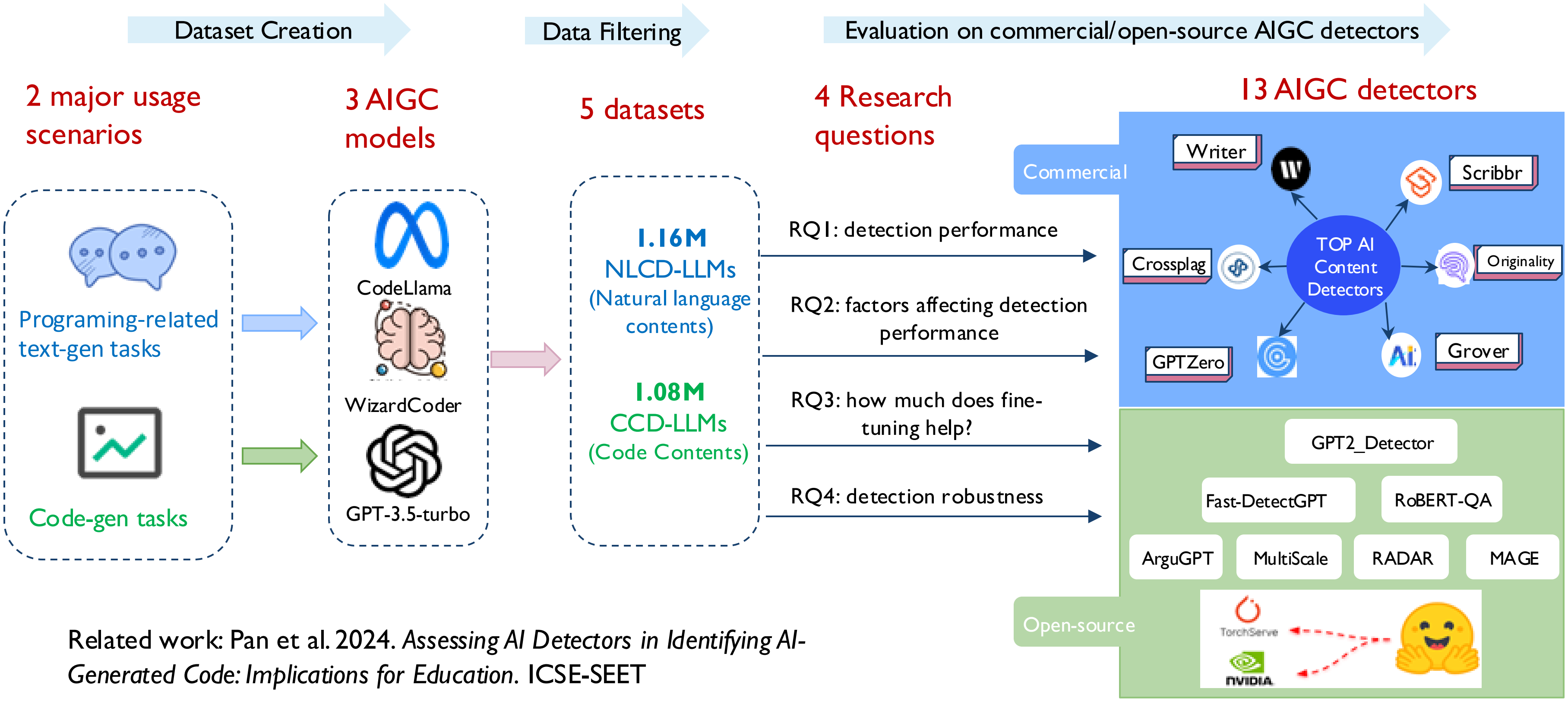
I'd say we prefer to talk to humans and we don't appreciate robots and fully automated tools. We also don't appreciate being used as validation to tools being developed out there. We're not free data collection agents for your automated tools.

I'm going to close this issue and mark it as spam, and will block the user since it's clearly not a known human. Feel free to open a PR fixing issues if you find any, using a normal human backed account.

adrinjalali closed this as not planned on Jul 10

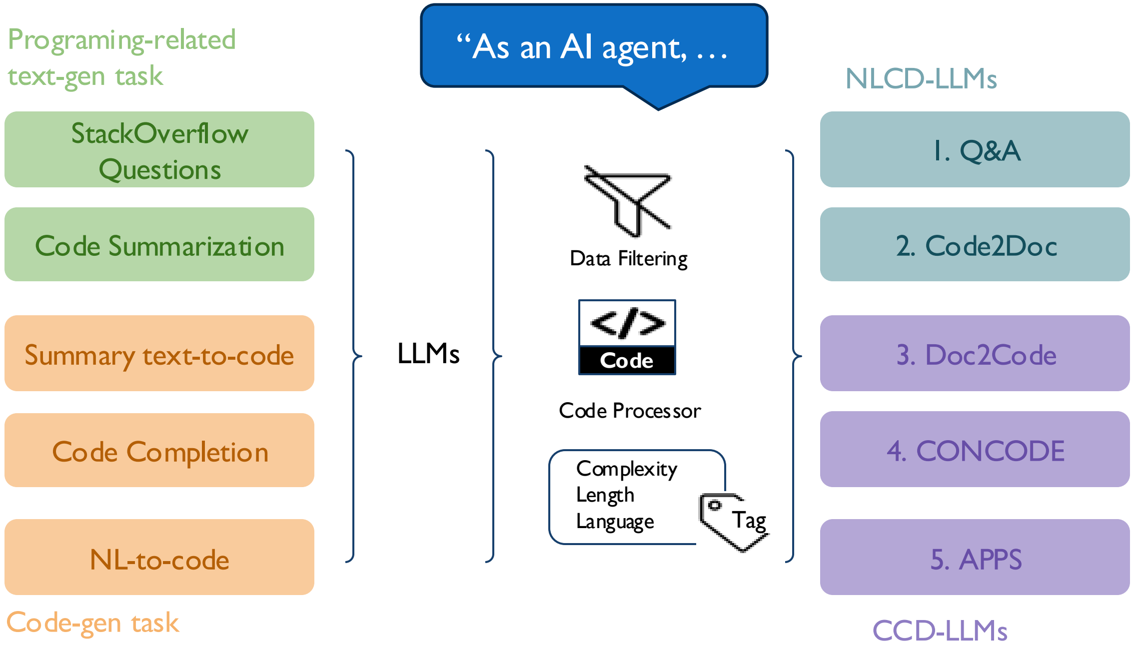
Source: https://github.com/scikit-learn/scikit-learn/issues/29440

An Empirical Study to Evaluate AIGC Detectors on Code Content



Related work: Pan et al. 2024. *Assessing AI Detectors in Identifying AI-Generated Code: Implications for Education.* ICSE-SEET

Dataset Creation



Training	Test
148K	2K
908K	97K
904K	96.6K
65K	1.4K
5K	3.7K
2.03M	0.2M

Dataset Creation: Programming-Related Q&A

Stack Exchange[1] dataset (coding topics)

Programing-related text-gen Task

StackOverflow Questions

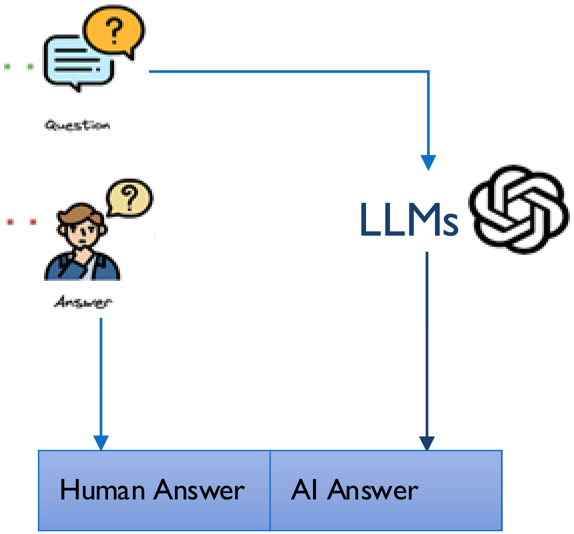
Code Summarization

Summary text-to-code

Code Completion

NL-to-code

Code-gen Task



1. Q&A

Training set: 148K

Test set: 2K

[1] https://github.com/EleutherAI/stackexchange-dataset

Dataset Creation: Code-to-Text

2. Code2Doc

Training set: 908K
Test set: 97K

CodeSearchNet[2] dataset

Programing-related text-gen Task

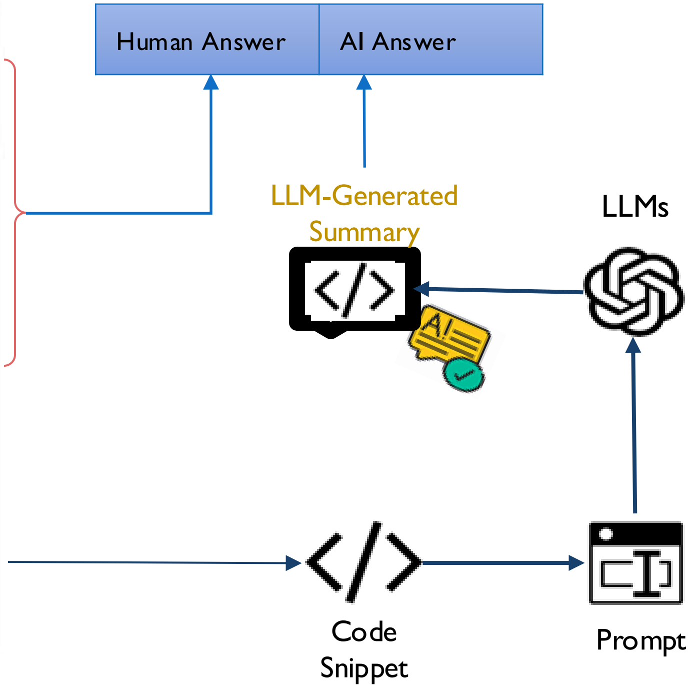
- StackOverflow Questions
- **Code Summarization**
- Summary text-to-code
- Code Completion
- NL-to-code

Code-gen Task

```python
def simplify(x, n):
    """
    This function takes two string representations of fractions,
    x and n, and returns True if x * n
    evaluates to a whole number and False otherwise.

    Both x and n are in the format of <numerator>/<denominator>,
    where both numerator and denominator
    are positive whole numbers.

    The function simplifies the expression x * n by converting
    the string representations to integers,
    multiplying them, and checking if the result is a whole
    number.

    simplify("1/5", "5/1") = True
    simplify("1/6", "2/1") = False
    simplify("7/10", "10/2") = False
    """
```

code snippet

```python
    # Convert the string representations to integers
    x_numerator, x_denominator = map(int, x.split('/'))
    n_numerator, n_denominator = map(int, n.split('/'))

    # Multiply the fractions
    result_numerator = x_numerator * n_numerator
    result_denominator = x_denominator * n_denominator

    # Check if the result is a whole number
    return result_numerator % result_denominator == 0
```



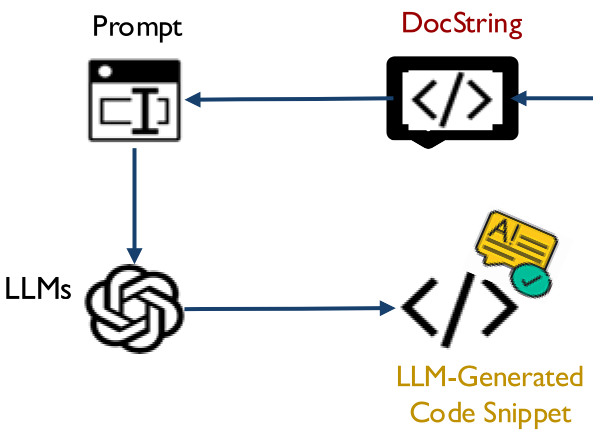
[2] https://github.com/github/CodeSearchNet

Dataset Creation: Text-to-Code

Programing-related text-gen Task

- StackOverflow Questions
- Code Summarization
- Summary text-to-code
- Code Completion
- NL-to-code

Code-gen Task



AI Answer	Human Answer

3. Doc2Code

Training set: 904K
Test set: 96.6K

CodeSearchNet dataset

```python
def simplify(x, n):
    """
    This function takes two string representations of fractions,
    x and n, and returns True if x * n
    evaluates to a whole number and False otherwise.

    Both x and n are in the format of <numerator>/<denominator>,
    where both numerator and denominator
    are positive whole numbers.

    The function simplifies the expression x * n by converting
    the string representations to integers,
    multiplying them, and checking if the result is a whole
    number.

    simplify("1/5", "5/1") = True
    simplify("1/6", "2/1") = False
    simplify("7/10", "10/2") = False
    """
```

Raw Code snippet

```python
    # Convert the string representations to integers
    x_numerator, x_denominator = map(int, x.split('/'))
    n_numerator, n_denominator = map(int, n.split('/'))

    # Multiply the fractions
    result_numerator = x_numerator * n_numerator
    result_denominator = x_denominator * n_denominator

    # Check if the result is a whole number
    return result_numerator % result_denominator == 0
```

RQ1: performance of existing detectors

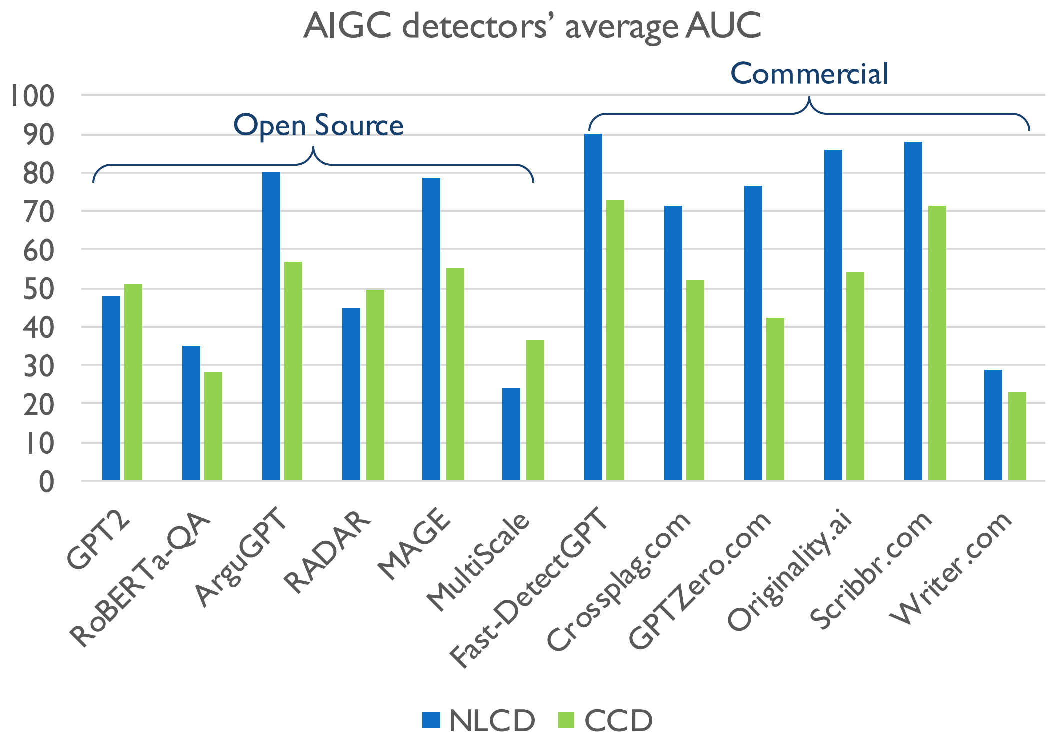
Detection accuracy for text-gen :

- Open-source tools (Avg. 57.26%) perform worse than commercial tools (Avg. 66.31%)
- Most commercial tools have good performance

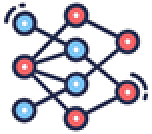
Detection accuracy for code-gen:

- Open-source tools (Avg. 50.25%) are slightly better than commercial tools (Avg. 48.90%)
- Most tools do not detect AI-gen code well, close to random guesses

Overview, average AUC for code-gen is 11.82% lower than text-gen



RQ2: factors affecting detection performance



Function complexity

Length

6 Programming languages

Table 4: The AUC performance of different detectors on the Doc2Code-LLM testset in terms of code complexity.

Complexity	Model	Avg.	GPT2_Detector	RoBETa-QA	ArguGPT	MAGE	RADAR	MultiScale	Fast-DetectGPT	GPTZero.me	Writer.com	Scribbr.com	Crossplag.com	Originality.ai
Easy	GPT-3.5-Turbo	50.82	50.36	48.58	45.69	51.36	44.01	40.55	57.91	50.77	47.39	**64.52**	50.26	58.40
	WizardCoder-15B	50.74	50.27	48.06	45.77	51.33	43.75	40.20	58.21	50.68	47.42	**64.63**	50.27	58.30
	CodeLlama-34B-Instruct	50.74	50.37	48.00	46.12	51.55	43.63	40.30	58.13	50.60	47.19	**64.62**	50.22	58.10
Medium	GPT-3.5-Turbo	49.12	49.83	33.34	54.19	49.01	48.76	40.14	56.02	49.66	45.37	**56.35**	50.40	56.35
	WizardCoder-15B	49.13	49.60	33.78	54.11	48.72	48.97	39.22	56.09	49.76	45.54	**56.66**	50.59	56.48
	CodeLlama-34B-Instruct	49.15	49.66	34.33	53.66	48.42	48.11	38.21	56.77	49.87	45.83	**57.15**	50.77	57.04
Hard	GPT-3.5-Turbo	48.45	48.77	35.42	53.63	47.37	43.19	35.06	**59.00**	49.65	47.44	50.34	52.73	58.76
	WizardCoder-15B	48.70	48.94	35.67	53.35	47.61	44.67	36.51	58.19	49.88	47.38	51.36	52.31	**58.53**
	CodeLlama-34B-Instruct	48.80	48.94	35.74	53.27	47.14	44.39	36.62	58.68	49.93	47.71	50.85	52.71	**59.57**
Avg.			49.64	39.22	51.09	49.17	45.50	38.53	57.67	50.09	46.81	57.39	51.14	57.95

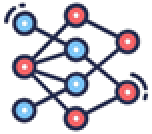
Table 5: The AUC performance of different detectors on the Doc2Code-LLM testset in terms of code length.

Length	Model	Avg.	GPT2_Detector	RoBETa-QA	ArguGPT	MAGE	RADAR	MultiScale	Fast-DetectGPT	GPTZero.me	Writer.com	Scribbr.com	Crossplag.com	Originality.ai
Short[0-72]]	GPT-3.5-Turbo	33.69	33.46	36.26	33.41	34.12	36.09	**36.55**	31.40	33.33	32.92	32.33	32.90	31.56
	WizardCoder-15B	50.89	50.87	50.08	49.49	52.90	48.42	46.70	53.50	49.84	45.21	**60.04**	49.81	53.82
	CodeLlama-34B-Instruct	50.68	51.08	49.36	49.91	52.87	47.94	46.17	53.56	49.78	44.64	**59.20**	49.81	53.85
Medium[72-197]	GPT-3.5-Turbo	32.87	33.39	32.08	32.81	33.02	31.80	32.16	33.13	33.32	32.88	**33.49**	33.09	33.25
	WizardCoder-15B	49.66	49.87	41.77	49.14	49.64	42.82	37.64	57.87	50.40	46.87	**60.75**	50.44	58.69
	CodeLlama-34B-Instruct	49.60	49.81	41.47	48.69	49.40	43.06	37.86	57.39	50.44	46.70	**61.04**	50.41	58.89
Long[197+]	GPT-3.5-Turbo	33.44	33.15	31.66	33.78	32.86	32.11	31.29	**35.47**	33.36	34.21	34.18	34.01	35.19
	WizardCoder-15B	50.39	49.03	37.54	49.40	47.73	41.59	33.90	63.23	50.85	50.15	62.80	52.82	**65.65**
	CodeLlama-34B-Instruct	49.96	49.15	38.40	49.69	48.41	41.97	34.59	61.48	50.51	49.02	61.17	52.14	**62.94**
Avg.			44.42	39.85	44.04	44.55	40.64	37.43	49.67	44.65	42.51	51.67	45.05	50.43

Table 6: The AUC performance of different detectors on the Doc2Code-LLM testset on 6 programming languages.

Language	Avg.	GPT2_Detector	RoBETa-QA	ArguGPT	MAGE	RADAR	MultiScale	Fast-DetectGPT	GPTZero.me	Writer.com	Scribbr.com	Crossplag.com	Originality.ai
Go	47.95	46.60	17.50	39.93	48.86	58.61	48.76	**59.82**	50.06	48.21	53.24	52.32	51.51
Java	51.75	50.31	33.78	59.94	52.37	47.12	36.36	63.55	51.82	47.96	**67.58**	50.42	59.80
Javascript	51.56	48.56	53.30	51.58	45.73	40.47	27.23	55.17	49.61	47.67	**80.91**	51.21	67.29
PHP	50.74	49.36	53.29	38.00	50.75	40.48	38.21	59.58	50.79	49.80	**68.13**	50.55	59.99
Python	47.88	52.83	45.54	**58.17**	49.52	41.79	39.91	57.38	48.17	41.70	37.03	50.76	51.80
Ruby	49.98	49.80	**87.28**	45.23	48.04	27.59	33.14	50.06	50.47	40.45	61.95	44.88	60.87
Avg.		49.58	48.45	48.81	49.21	42.68	37.27	57.59	50.15	45.96	61.47	50.02	58.54

RQ2: factors affecting detection performance



Function complexity

Length

6 Programming languages

TL;DR:

- More complex code seems more challenging to detect, especially for open-source detectors
- Longer code is easier to detect
- Most tools are stable across different languages, some tools struggle on one/two languages

RQ3: how much does fine-tuning help?

Fine-tuning can significantly improve detection performance

Table 7: Results of fine-tuned models on different *NLCD-Train* datasets.

Detector		NLCD-Test Q&A-LLM AUC	Q&A-LLM FPR	Q&A-LLM FNR	Code2Doc-LLM AUC	Code2Doc-LLM FPR	Code2Doc-LLM FNR
Unfined-tuned RoBERTa-QA		0.37	0.07	0.91	0.34	0.37	0.71
RoBERTa-QA	Q&A-LLM	1.00	0. 00	0.00	0.69	0.99	0.00
	Code2Doc-LLM	0.84	0.08	0.41	1.00	0.00	0.00
	Composite-NL	1.00	0.00	0.00	1.00	0.00	0.00
	Avg.	0.95	0.03	0.14	0.90	0.33	0.00
MLP	Q&A-LLM	0.89	0.21	0.18	0.42	0.34	0.74
	Code2Doc-LLM	0.43	0.52	0.65	1.00	0.00	0.01
	Composite-NL	1.00	0.01	0.01	0.78	0.34	0.24
	Avg.	0.77	0.25	0.28	0.73	0.23	0.33

AUC after fine-tuning

Text

92.5%

Code

78.3%

Table 8: Results of fine-tuned models on different *CCD-Train* datasets.

Detector		CCD-Test CONCODE-LLM AUC	CONCODE-LLM FPR	CONCODE-LLM FNR	Doc2Code-LLM AUC	Doc2Code-LLM FPR	Doc2Code-LLM FNR	APPS-LLM AUC	APPS-LLM FPR	APPS-LLM FNR
Unfined-tuned RoBERTa-QA		0.03	1.00	0.04	0.43	0.56	0.53	0.38	0.62	0.51
RoBERTa-QA	APPS-LLM	0.30	0.00	1.00	0.61	0.39	0.43	0.94	0.49	0.00
	CONCODE-LLM	1.00	0.00	0.00	0.53	0.99	0.00	0.52	1.00	0.00
	Doc2Code-LLM	0.94	0.00	0.94	1.00	0.04	0.01	0.56	0.80	0.06
	Composite-Code	1.00	0.00	0.00	1.00	0.04	0.01	0.84	0.53	0.01
	Avg.	0.86	0.00	0.49	0.78	0.37	0.11	0.71	0.70	0.02
MLP	APPS-LLM	0.29	0.00	1.00	0.51	0.13	0.83	0.73	0.38	0.29
	CONCODE-LLM	0.99	0.02	0.08	0.44	0.98	0.02	0.43	0.68	0.40
	Doc2Code-LLM	0.66	0.48	0.34	0.89	0.22	0.17	0.56	0.45	0.47
	Composite-Code	0.98	0.06	0.13	0.89	0.24	0.16	0.68	0.41	0.32
	Avg.	0.73	0.14	0.38	0.68	0.40	0.30	0.60	0.48	0.37

TL;DR: Fine-tuning can significantly improve the performance of the detectors for detecting code contents

RQ4: detection robustness under mutations

Mutations	Descriptions	Example before transformation	Example after transformation

FuncAddLine	Equivalent transformation between a constant or a newline assigned by same constant.	`f(a, b, c)`	`f(`
			`a,`
			`b, c)`

For2While	Equivalent transformation among for structure and while structure.	`For (i in range(9)) :`
		`    Body;`
			`i=0;`
			`while i<9 :`
			`    Body;`

AugAssign	Equivalent numerical calculation transformation, e.g., ++, --, +=, &=, |=	`a += 1`	`a = a + 1`

AddDeadCode	Insert some dead code fragments, unused statements or repeated statements in the code.	`class work:`
		`    pass`
			`class Foo:  # noqa: DC03`
			`    pass`
			`class work:`
			`    pass`

VarRename	Rename the function names and variable names with all their occurrence with newly generated names such as F0, V1, V2	`def func1(var1):` `    pass`	`def func_new1(var_new1):` `    pass`

TL;DR: the mutation operators make it harder to detect AI-generated code, but they do not have significant impact on detecting human-written contents

Summary

yi_li@ntu.edu.sg
@liyistc