Fine-Tuning LLM for Higher-Order Code Generation In COCOBOTS domain

Yin-Chien Pai (Eszter) | Cognitive Systems | University of Potsdam

First-Order Code vs. Higher-Order Code

- Natural Language Instruction:
  - We are building the M5 nut assembly. Stack a nut and washer in the 5th row and 3rd column. Use red for the nut and yellow for the washer.

First-Order Code:

```
put(board, “nut”, “red”, 4, 3)

put(board, “washer”, “yellow”, 4, 3)
```

Higher-Order Code:

```
def m5(board, colors, x, y):

   shapes = [“nut”, “washer”]

   for shape, color in zip(shapes, colors):

       put(board, shape, color, x, y)

m5(board, ["red", "yellow"], 4, 2)
```

Motivation

- The need for **accurate function generation** and the **limitations of current LLMs** in COCOBOTS domain.

- Improved accuracy in function generation can aid developers, reduce errors, and enhance productivity.

Objectives

- Have a smaller-sized (3b/7b/8b/13b) LLM **accurately generate higher-order code** from natural language instructions.

- Have the same model to be able to tackle/generate **both first-order code and higher-order code**

Evaluation Overview

Models

- Codellama-Instruct 7B/13B
- Llama 3.1 -8B
- Llama 3.2 -3B
- Mistral 7B
- Stable Code 3B

Dataset

First-Order	Train (4144), Validation(500) and Test (500)
Higher-Order	Train (1072), Validation(130) and Test (130)

Metrics

- Exact Match (EM)
- CodeBLEU
- Execution Success (ES)

Results - First-Order Training, Inference on Both

Models	Exact match				CodeBLEU				Execution Success			
	Pretrained		Fine-Tuned		Pretrained		Fine-Tuned		Pretrained		Fine-Tuned	
	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order
CodeLlama-7b-Instruct	0.00%	0.00%	47.20%	0.00%	29.04%	12.77%	78.96%	16.44%	0.00%	0.00%	**91.40%**	14.60%
CodeLlama-13b Instruct	0.00%	0.00%	9.80%	0.00%	27.08%	13.54%	69.81%	16.22%	0.00%	0.00%	84.80%	11.50%
Llama-3.1-8B	0.00%	0.00%	**54.00%**	0.00%	25.10%	12.59%	**80.89%**	13.42%	0.00%	0.00%	**100%**	76.92%
Llama-3.2-3B	0.00%	0.00%	44.40%	0.00%	25.48%	12.81%	76.97%	29.08%	0.00%	0.00%	81%	4.60%
Mistral-7B-v0.1	0.00%	0.00%	**100.00%**	0.00%	25.44%	12.86%	**100.00%**	10.68%	0.00%	0.00%	**100%**	65.38%

* Unsloth qunatized fine-tuning

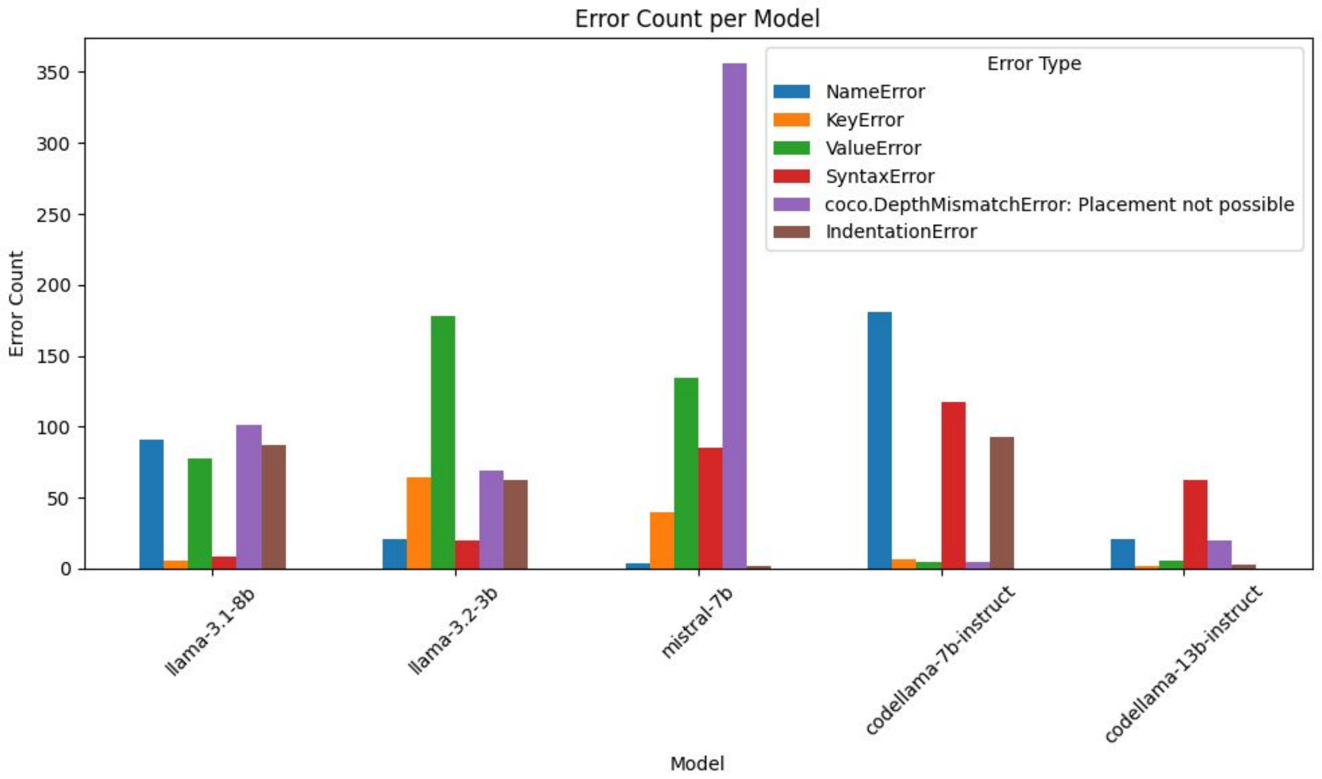
Results - Higher Order Training, Inference on Both

	Exact match				CodeBLEU				Execution Success			
	Pretrained		Fine-Tuned		Pretrained		Fine-Tuned		Pretrained		Fine-Tuned	
	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order
CodeLlama-7b-Instruct	0.00%	0.00%	2.60%	7.69%	29.04%	12.77%	47.27%	**72.76%**	0.00%	0.00%	14.60%	41.50%
CodeLlama-13b Instruct	0.00%	0.00%	0.00%	0.00%	27.08%	13.54%	**67.25%**	23.09%	0.00%	0.00%	**89.20%**	11.50%
Llama-3.1-8B	0.00%	0.00%	0.00%	26.15%	25.10%	12.59%	**52.76%**	**81.66%**	0.00%	0.00%	14.80%	**83.80%**
Llama-3.2-3B	0.00%	0.00%	13.00%	11.54%	25.48%	12.81%	**57.83%**	60.93%	0.00%	0.00%	24%	23.80%
Mistral-7B-v0.1	0.00%	0.00%	0.00%	0.00%	25.44%	12.86%	36.90%	58.33%	0.00%	0.00%	0%	**65.38%**

Results - Both Training, Inference on Both

	Exact Match				CodeBLEU				Exec Success			
	Pretrained		Fine-Tuned		Pretrained		Fine-Tuned		Pretrained		Fine-Tuned	
	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order
CodeLlama-7b Instruct	0.00%	0.00%	**95.40%**	44.62%	29.04%	12.77%	**98.36%**	**90.57%**	0.00%	0.00%	**95.40%**	54.60%
CodeLlama-13b Instruct	0.00%	0.00%	**92.40%**	6.92%	27.08%	13.54%	**98.74%**	34.41%	0.00%	0.00%	**92.40%**	**69.20%**
Llama-3.1-8B	0.00%	0.00%	54.00%	3.08%	25.10%	12.59%	80.89%	36.52%	0.00%	0.00%	**100%**	22.30%
Llama-3.2-3B	0.00%	0.00%	54.00%	19.23%	25.48%	12.81%	80.89%	81.68%	0.00%	0.00%	**100%**	53%
Mistral-7B-v0.1	0.00%	0.00%	**100.00%**	4.62%	25.44%	12.86%	**100.00%**	74.25%	0.00%	0.00%	**100%**	27.69%

Error Analysis: Error Counts per model



Syntax error example

Input Instruction:

These are the instructions to build bns.
Place a red bridge horizontally in the 7th row, 4th column.
Place a blue nut in the 7th row, 4th column.
Place a red screw in the 7th row, 4th column.

Output:

```
bns(board, (’red’, ’blue’, ’red’)
```

SyntaxError: ’(’ was never closed:

Placement error example -

Input Instruction:
place a green bridge horizontally in the 6th row, 1st column

- The ground truth code is:

```
put(board, 'bridge-h', 'green', 5, 0)
```

- The generated code is:

```
def shapes(board, shapes, colors, x, y):
   for shape, color, dx, dy in zip(shapes, colors,[0, 0, 0, 0], [0, 1, 0, 0]):
      put(board, shape, color, x + dx, y + dy)

shapes(board, ('bridge-h', 'bridge-v', 'bridge-h', 'bridge-v'),
('green', 'red', 'blue', 'yellow'), 5, 0)
```

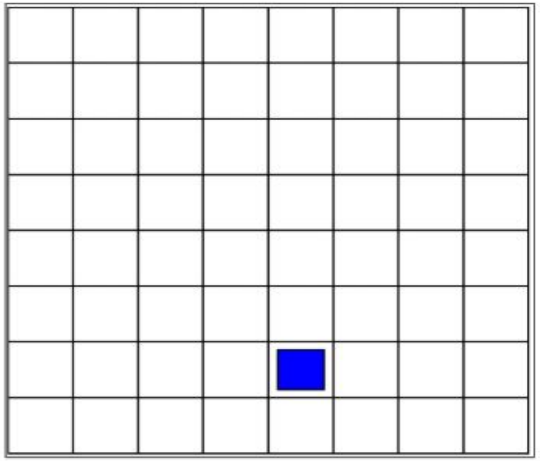
DimensionsMismatchError:
Raised when the dimensions of the board do not match the dimensions of input x,y

Placement error example

Input Instruction:
place a red nut in the 7th row, 5th column

Ground Truth

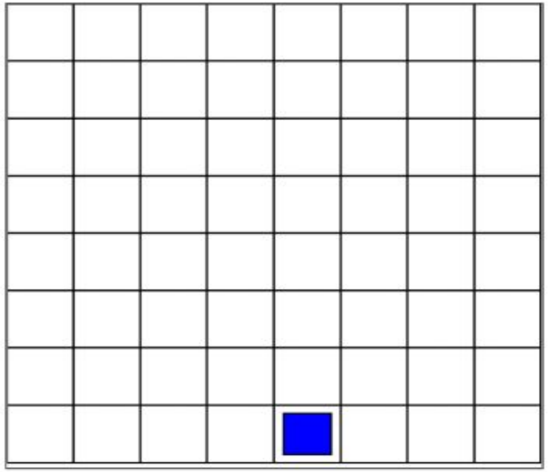
```
put(board, “nut”, “red”, 6, 4)
```



Generated

```
put(board, “nut”, “red”, 7, 4)
```

The row number (7) is wrong



Conclusion

- First-order code inference is high (~80-100%) (irrespective of the training data)

- Higher-order code inference is high when the models trained only on higher-order data (83%) and the performance dips with the training only on first-order code (76%) and combined data (69%)

- Syntax Errors (parenthesis not closed) and Placement Errors are the most common error types

BACKUP

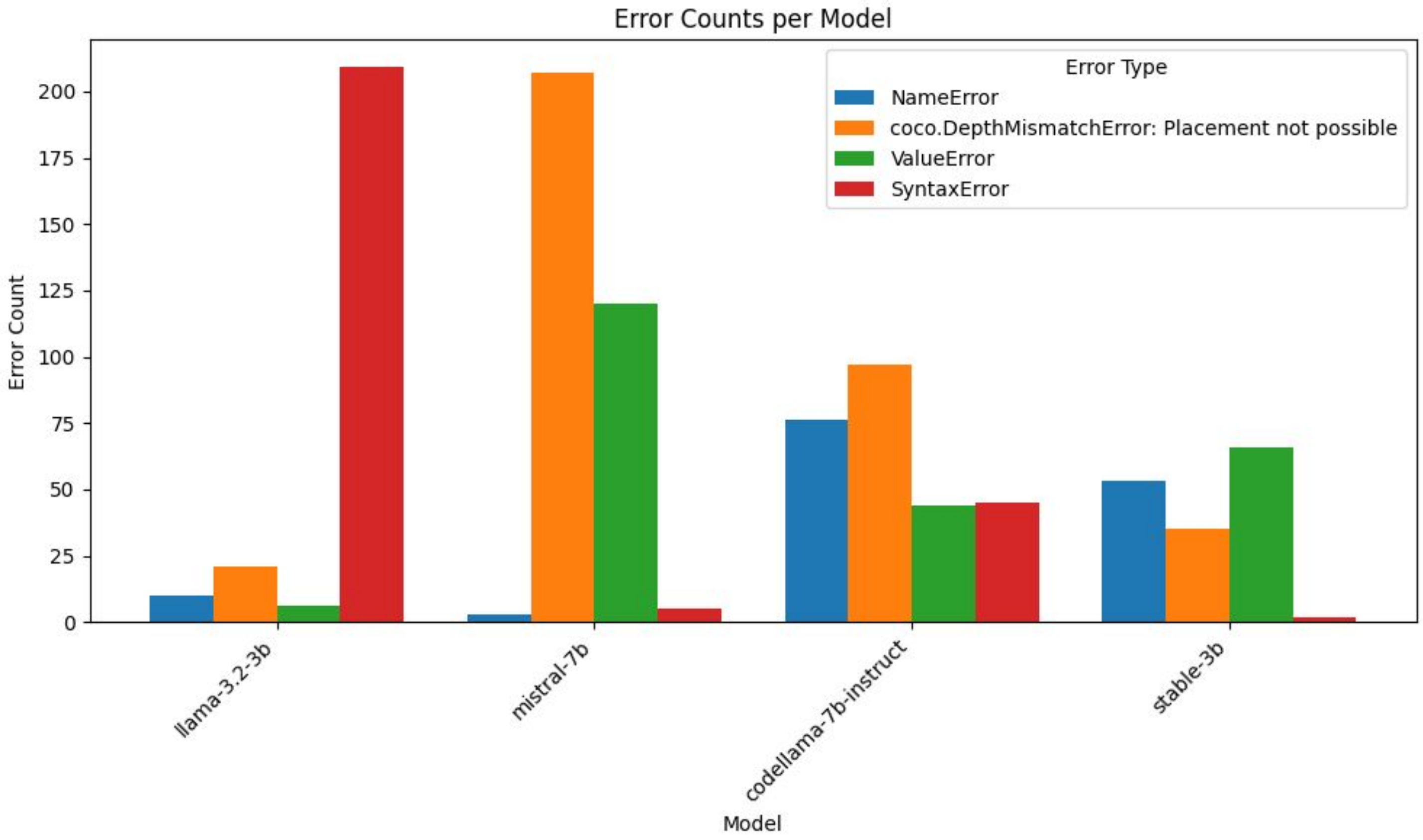
LoRA Results (first order training)

	exact match				codebleu				exec success			
	Pretrained		Fine-Tuned		Pretrained		Fine-Tuned		Pretrained		Fine-Tuned	
	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order
CodeLlama-7b-Instruct	0.00%	0.00%	54.00%	0.00%	29.04%	12.77%	80.89%	17.70%	0.00%	0.00%	100%	33%
Llama-3.1-8B	0.00%	0.00%	54.00%	0.00%	25.10%	12.59%	80.89%	30.50%	0.00%	0.00%	100%	0.00%
Llama-3.2-3B	0.00%	0.00%	53.20%	0.00%	25.48%	12.81%	80.58%	21.09%	0.00%	0.00%	98.80%	36.92%
Mistral-7B-v0.1	0.00%	0.00%	54.00%	0.00%	25.44%	12.86%	80.89%	30.48%	0.00%	0.00%	100%	0.00%
Stable-Code-3B	0.00%	0.00%	0.00%	0.40%	27.11%	13.97%	61.41%	16.61%	0.00%	0.00%	0.80%	0.00%

LoRA Results (higher order training)

	exact match				codebleu				exec success			
	Pretrained		Fine-Tuned		Pretrained		Fine-Tuned		Pretrained		Fine-Tuned	
	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order	First-Order	Higher-Order
CodeLlama-7b-Instruct	0.00%	0.00%	0.00%	0.00%	29.04%	12.77%	36.49%	71.70%	0.00%	0.00%	5.80%	90.00%
Llama-3.2-3B	0.00%	0.00%	0.00%	0.00%	25.48%	12.81%	36.21%	74.41%	0.00%	0.00%	52.60%	67.70%
Mistral-7B-v0.1	0.00%	0.00%	0.00%	1.54%	25.44%	12.86%	35.96%	78.19%	0.00%	0.00%	1.40%	97.70%
Stable-Code-3B	0.00%	0.00%	0.00%	0.00%	27.11%	13.97%	37.02%	75.03%	0.00%	0.00%	4.40%	99.20%

Error Analysis: Error Counts per model (without Unsloth)



Error Categories - Name Error

Input Instruction:

These are the instructions to build wwb.
Place a green washer in the 8th row, 1st column.
Place a yellow washer in the 8th row, 2nd column.
Place a red bridge horizontally in the 8th row, 1st column.

- **NameError: name 'shapes' is not defined:**

Output:

```
def wwb(board, shapes, colors, x, y):
        shapes = ['washer', 'washer', 'bridge-h']
        colors = ['green', 'yellow','red']
        for shape, color, dx, dy in zip(shapes, colors, [0, 0, 0], [0, 1, 0]):
                put(board, shape, color, x + dx, y + dy)

wwb(board, shapes, colors, 7, 0)
```

Error Categories - Value Error

Input Instruction:
place a red bridge vertically in the 7th row, 2nd column

- **ValueError: not enough values to unpack (expected 4, got 3)**

```
def bv(board, color, x, y):
    shapes = ['bridge-v']
    for shape, color, dx, dy in zip(shapes, [0, 0], [0, 0]):
        put(board, shape, color, x + dx, y + dy)

bv(board,'red', 6, 1)
```