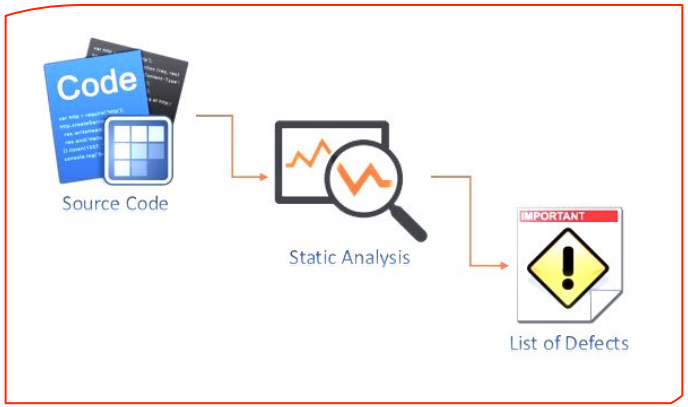
A Static Evaluation of Code Completion by Large Language Models

- Ding et al. 2023



CSCI 544 Group **37**: Anuroop, Indrani, Abhishek, Kayvan, Vishesh

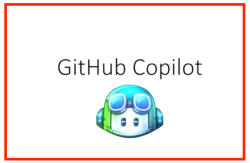
10/05/2023

Table of contents

01

Background

- Code generation has become a popular application of LLMs in recent times



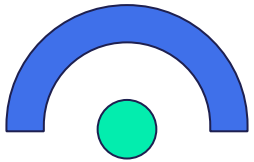
- Such tools aim to enhance developer productivity and shorten development time

02

Motivation

- **Evaluation** of generated code is necessary
- About 70% model generated code is discarded (Ziegler et al., 2022)
- **Execution-based benchmarks** though useful are quite expensive and difficult to scale
- **Static analysis tools** analyze programs without the need for execution and are thus faster

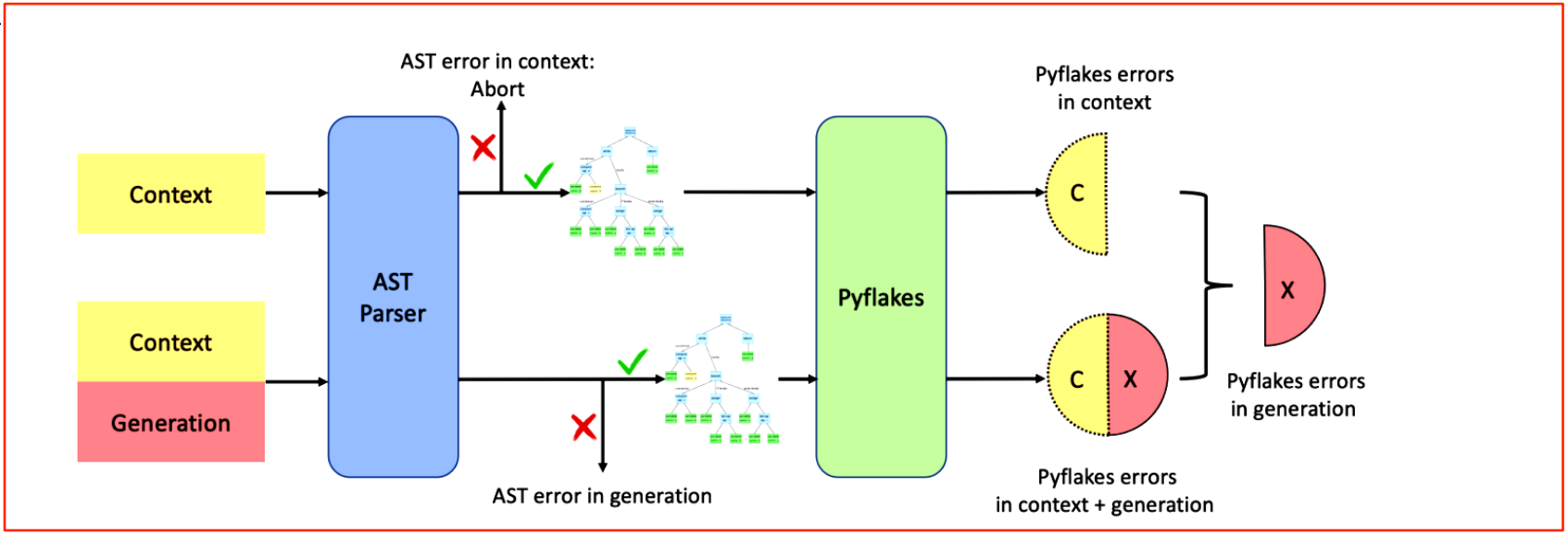
```
def calculate_sum(numbers):
    total = 0
    for num in numbers:
        total += value
    return total

numbers_list = [1, 2, 3, 4, 5]
result = calculate_sum(numbers_list)
print("Total sum is:", result)
```



03

Main Idea

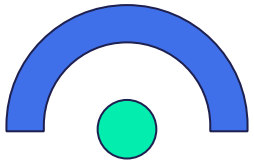
Develop a **static evaluation framework** to quantify static errors in Python code completions, by leveraging **Abstract Syntax Trees**



04

Experimental Setup

- **Language:** Python
- **AST Generator:** Tree Sitter
- **Linter:** PyFlakes
- **Dataset:** Publicly available code sourced from GitHub repositories (100K problems)
- **LLM:** CodeGen



05

Results

AST error statistics and findings

Model	Temp	Total	EOF	Non EOF	Invalid Syntax	"print" Missing Parentheses	Keyword Argument Repeated
CodeGen-16B	0.4	7.330%	7.236%	0.094%	0.042%	0.041%	0.004%
CodeGen-6B		7.446%	7.253%	0.193%	0.081%	0.094%	0.006%
CodeGen-2B		7.272%	7.177%	0.095%	0.052%	0.018%	0.008%
CodeGen-350M		8.703%	8.593%	0.110%	0.041%	0.016%	0.028%
CodeGen-2B	0.2	8.067%	7.982%	0.085%	0.045%	0.018%	0.008%
	0.4	7.272%	7.177%	0.095%	0.052%	0.018%	0.008%
	0.6	6.823%	6.713%	0.110%	0.060%	0.020%	0.008%
	0.8	7.496%	7.337%	0.159%	0.085%	0.029%	0.014%

- Codes generated by models, unless incomplete, are **mostly parsable into ASTs**, regardless of model size or temperature
- **Interpreter version mismatch** is one of the major reasons for non-EOF AST errors

05

Results

AST error statistics and findings

Model	Temp	Total	EOF	Non EOF	Invalid Syntax	"print" Missing Parentheses	Keyword Argument Repeated
CodeGen-16B	0.4	7.330%	7.236%	0.094%	0.042%	0.041%	0.004%
CodeGen-6B		7.446%	7.253%	0.193%	0.081%	0.094%	0.006%
CodeGen-2B		7.272%	7.177%	0.095%	0.052%	0.018%	0.008%
CodeGen-350M		8.703%	8.593%	0.110%	0.041%	0.016%	0.028%
CodeGen-2B	0.2	8.067%	7.982%	0.085%	0.045%	0.018%	0.008%
	0.4	7.272%	7.177%	0.095%	0.052%	0.018%	0.008%
	0.6	6.823%	6.713%	0.110%	0.060%	0.020%	0.008%
	0.8	7.496%	7.337%	0.159%	0.085%	0.029%	0.014%

- Codes generated by models, unless incomplete, are **mostly parsable into ASTs**, regardless of model size or temperature
- **Interpreter version mismatch** is one of the major reasons for non-EOF AST errors

PyFlakes error statistics and findings

Model	Temp	Undefined Name	Unused Variable	FString Missing Placeholders	Unused Import	Redefined While Unused	Undefined Local
CodeGen-16B	0.4	4.323%	1.729%	0.135%	0.107%	0.131%	0.047%
CodeGen-6B		4.374%	1.775%	0.089%	0.149%	0.126%	0.055%
CodeGen-2B		4.364%	1.810%	0.147%	0.150%	0.146%	0.065%
CodeGen-350M		4.472%	2.032%	0.151%	0.173%	0.155%	0.095%
CodeGen-2B	0.2	4.206%	1.751%	0.125%	0.139%	0.139%	0.067%
	0.4	4.364%	1.810%	0.147%	0.150%	0.146%	0.065%
	0.6	4.711%	2.000%	0.188%	0.170%	0.159%	0.076%
	0.8	5.377%	2.490%	0.240%	0.247%	0.184%	0.086%

- **Undefined Name** and **Unused Variable** are the most common error types
- Higher temperature always leads to more errors of every type
- While larger models are more accurate code generators, scaling-up model size does not always reduce errors
- Significant correlation between errors in context and errors in generation

PyFlakes error statistics and findings

Model	Temp	Undefined Name	Unused Variable	FString Missing Placeholders	Unused Import	Redefined While Unused	Undefined Local
CodeGen-16B	0.4	4.323%	1.729%	0.135%	0.107%	0.131%	0.047%
CodeGen-6B		4.374%	1.775%	0.089%	0.149%	0.126%	0.055%
CodeGen-2B		4.364%	1.810%	0.147%	0.150%	0.146%	0.065%
CodeGen-350M		4.472%	2.032%	0.151%	0.173%	0.155%	0.095%
CodeGen-2B	0.2	4.206%	1.751%	0.125%	0.139%	0.139%	0.067%
	0.4	4.364%	1.810%	0.147%	0.150%	0.146%	0.065%
	0.6	4.711%	2.000%	0.188%	0.170%	0.159%	0.076%
	0.8	5.377%	2.490%	0.240%	0.247%	0.184%	0.086%

- **Undefined Name** and **Unused Variable** are the most common error types
- **Higher temperature** always leads to more errors of every type
- While larger models are more accurate code generators, scaling-up model size does not always reduce errors
- Significant correlation between errors in context and errors in generation

PyFlakes error statistics and findings

Model	Temp	Undefined Name	Unused Variable	FString Missing Placeholders	Unused Import	Redefined While Unused	Undefined Local
CodeGen-16B		4.323%	1.729%	0.135%	0.107%	0.131%	0.047%
CodeGen-6B	0.4	4.374%	1.775%	0.089%	0.149%	0.126%	0.055%
CodeGen-2B	0.4	4.364%	1.810%	0.147%	0.150%	0.146%	0.065%
CodeGen-350M		4.472%	2.032%	0.151%	0.173%	0.155%	0.095%
CodeGen-2B	0.2	4.206%	1.751%	0.125%	0.139%	0.139%	0.067%
	0.4	4.364%	1.810%	0.147%	0.150%	0.146%	0.065%
	0.6	4.711%	2.000%	0.188%	0.170%	0.159%	0.076%
	0.8	5.377%	2.490%	0.240%	0.247%	0.184%	0.086%

- **Undefined Name** and **Unused Variable** are the most common error types
- **Higher temperature** always leads to more errors of every type
- While larger models are more accurate code generators, **scaling-up model size does not always reduce errors**
- Significant correlation between errors in context and errors in generation

PyFlakes error statistics and findings

Model	Temp	Undefined Name	Unused Variable	FString Missing Placeholders	Unused Import	Redefined While Unused	Undefined Local
CodeGen-16B		4.323%	1.729%	0.135%	0.107%	0.131%	0.047%
CodeGen-6B	0.4	4.374%	1.775%	0.089%	0.149%	0.126%	0.055%
CodeGen-2B		4.364%	1.810%	0.147%	0.150%	0.146%	0.065%
CodeGen-350M		4.472%	2.032%	0.151%	0.173%	0.155%	0.095%
CodeGen-2B	0.2	4.206%	1.751%	0.125%	0.139%	0.139%	0.067%
	0.4	4.364%	1.810%	0.147%	0.150%	0.146%	0.065%
	0.6	4.711%	2.000%	0.188%	0.170%	0.159%	0.076%
	0.8	5.377%	2.490%	0.240%	0.247%	0.184%	0.086%

- **Undefined Name** and **Unused Variable** are the most common error types
- **Higher temperature** always leads to more errors of every type
- While larger models are more accurate code generators, **scaling-up model size does not always reduce errors**
- Significant correlation between errors in context and errors in generation

PyFlakes error statistics and findings

Model	Temp	Undefined Name	Unused Variable	FString Missing Placeholders	Unused Import	Redefined While Unused	Undefined Local
CodeGen-16B	0.4	4.323%	1.729%	0.135%	0.107%	0.131%	0.047%
CodeGen-6B		4.374%	1.775%	0.089%	0.149%	0.126%	0.055%
CodeGen-2B		4.364%	1.810%	0.147%	0.150%	0.146%	0.065%
CodeGen-350M		4.472%	2.032%	0.151%	0.173%	0.155%	0.095%
CodeGen-2B	0.2	4.206%	1.751%	0.125%	0.139%	0.139%	0.067%
	0.4	4.364%	1.810%	0.147%	0.150%	0.146%	0.065%
	0.6	4.711%	2.000%	0.188%	0.170%	0.159%	0.076%
	0.8	5.377%	2.490%	0.240%	0.247%	0.184%	0.086%

- **Undefined Name** and **Unused Variable** are the most common error types
- **Higher temperature** always leads to more errors of every type
- While larger models are more accurate code generators, **scaling-up model size does not always reduce errors**
- Significant **correlation** between errors in **context** and errors in **generation**

06

Critique

- Work is restricted to Python, left-to-right code generation and limited by the capacity of linter
- Non-EOF AST errors may be higher for syntactically stricter languages such as C++ and Java
- Cross-file context, which is more relevant with modularization of code and reflects best practices, is not considered in this evaluation framework
- Results are highly specific to CodeGen and cannot be generalized to other LLMs in the same category
- Code quality evaluation is not considered

06

Critique

- Work is restricted to Python, left-to-right code generation and limited by the capacity of linter
- Non-EOF AST errors may be higher for syntactically stricter languages such as C++ and Java
- Cross-file context, which is more relevant with modularization of code and reflects best practices, is not considered in this evaluation framework
- Results are highly specific to CodeGen and cannot be generalized to other LLMs in the same category
- Code quality evaluation is not considered

06

Critique

- Work is restricted to Python, left-to-right code generation and limited by the capacity of linter
- Non-EOF AST errors may be higher for syntactically stricter languages such as C++ and Java
- Cross-file context, which is more relevant with modularization of code and reflects best practices, is not considered in this evaluation framework
- Results are highly specific to CodeGen and cannot be generalized to other LLMs in the same category
- Code quality evaluation is not considered

06

Critique

- Work is restricted to Python, left-to-right code generation and limited by the capacity of linter
- Non-EOF AST errors may be higher for syntactically stricter languages such as C++ and Java
- Cross-file context, which is more relevant with modularization of code and reflects best practices, is not considered in this evaluation framework
- Results are highly specific to CodeGen and cannot be generalized to other LLMs in the same category
- Code quality evaluation is not considered

06

Critique

- Work is restricted to Python, left-to-right code generation and limited by the capacity of linter
- Non-EOF AST errors may be higher for syntactically stricter languages such as C++ and Java
- Cross-file context, which is more relevant with modularization of code and reflects best practices, is not considered in this evaluation framework
- Results are highly specific to CodeGen and cannot be generalized to other LLMs in the same category
- Code quality evaluation is not considered

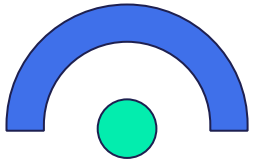
07

Future Work

1. Repeat work with different

   Linters: SonarLint
   Languages: C++, Java, JavaScript
   Models: Code Llama

2. Explore the severity of less frequent but critical AST and linter errors
3. Feed the statistics obtained from the evaluation framework back to the model
4. Increase the token limit number and re-explore evaluation statistics



Thank you

CSCI 544 Group **37**: Anuroop, Indrani, Abhishek, Kayvan, Vishesh