Large Language Models of Code Fail at Completing Code with Potential Bugs

Tuan Dinh, Jinman Zhao, Samson Tan, Renato Negrinho,
Leonard Lausen, Sheng Zha, and George Karypis

University of Wisconsin-Madison AWS AI Research and Education

NEURAL INFORMATION PROCESSING SYSTEMS

Code Completion with Code Language Models

Code Completion with Code Language Models

```python
""" You're given a list of deposit and withdrawal operations on a bank account that
    starts with zero balance. Your task is to detect if at any point the balance of
    account fallls below zero, and at that point function should return True.
    Otherwise it should return False."""
```

Problem statement

```python
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance += op
        if balance < 0:
```

Partial code

Code Completion with Code Language Models

```
""" You're given a list of deposit and withdrawal operations on a bank account that
    starts with zero balance. Your task is to detect if at any point the balance of
    account fallls below zero, and at that point function should return True.
    Otherwise it should return False."""
```

Problem statement

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance += op
        if balance < 0:
```

Partial code

```
            return True
    return False
```

Completion

Code Completion with Code Language Models

Code-LLMs achieve > 50% pass rate on various benchmarks

Problem statement

```
""" You're given a list of deposit and withdrawal operations on a bank account that
    starts with zero balance. Your task is to detect if at any point the balance of
    account fallls below zero, and at that point function should return True.
    Otherwise it should return False."""
```

Partial code

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance += op
        if balance < 0:
```

Completion

```
            return True
    return False
```

Code Completion with Code Language Models

Code-LLMs achieve > 50% pass rate on various benchmarks

```
""" You're given a list of deposit and withdrawal operations on a bank account that
    starts with zero balance. Your task is to detect if at any point the balance of
    account fallls below zero, and at that point function should return True.
    Otherwise it should return False."""
```

Problem statement

Existing completion models assume **error-free** inputs …

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance += op
        if balance < 0:
```

Partial code

```
            return True
    return False
```

Completion

Code Completion with Code Language Models

Code-LLMs achieve > 50% pass rate on various benchmarks

Existing completion models assume **error-free** inputs …

Bugs in code are inevitable!

(esp for the in-progress partial code)

Buggy-Code Completion

Motivated scenarios: typos or logical mistakes during coding process

Buggy-Code Completion

Motivated scenarios: typos or logical mistakes during coding process

Problem statement

```
""" You're given a list of deposit and withdrawal operations on a bank account that
    starts with zero balance. Your task is to detect if at any point the balance of
    account fallls below zero, and at that point function should return True.
    Otherwise it should return False."""
```

Partial code

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance += op
        if balance < 0:
```

Completion

```
            return True
    return False
```

Buggy-Code Completion

Motivated scenarios: typos or logical mistakes during coding process

```
""" You're given a list of deposit and withdrawal operations on a bank account that
    starts with zero balance. Your task is to detect if at any point the balance of
    account fallls below zero, and at that point function should return True.
    Otherwise it should return False."""
```

Problem statement

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance += op
        if balance < 0:
```

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance -= op
        if balance < 0:
```

Partial code

```
            return True
    return False
```

Completion

Buggy-Code Completion

Motivated scenarios: typos or logical mistakes during coding process

Problem statement

```
""" You're given a list of deposit and withdrawal operations on a bank account that
    starts with zero balance. Your task is to detect if at any point the balance of
    account fallls below zero, and at that point function should return True.
    Otherwise it should return False."""
```

Partial code

Without potential bugs:

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance += op
        if balance < 0:
```

With potential bugs:

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance -= op
        if balance < 0:
```

Completion

```
            return True
    return False
```

Without potential bugs

With potential bugs

A change from += → -= results in a **potential bug**

→ partial code + original completion fails test: below_zero(1, 2) == False

→ the code completion should change

Problem statement

```
""" You're given a list of deposit and withdrawal operations on a bank account that
    starts with zero balance. Your task is to detect if at any point the balance of
    account fallls below zero, and at that point function should return True.
    Otherwise it should return False."""
```

Partial code

Without potential bugs:

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance += op
        if balance < 0:
```

With potential bugs:

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance -= op
        if balance < 0:
```

Completion

Without potential bugs:

```
            return True
    return False
```

Without potential bugs

With potential bugs

A change from += → -= results in a **potential bug**

→ partial code + original completion fails test: below_zero(1, 2) == False

→ the code completion should change

Problem statement

```
""" You're given a list of deposit and withdrawal operations on a bank account that
    starts with zero balance. Your task is to detect if at any point the balance of
    account fallls below zero, and at that point function should return True.
    Otherwise it should return False."""
```

Partial code

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance += op
        if balance < 0:
```

Completion

```
            return True
    return False
```

Without potential bugs

Partial code

```
from typing import List
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for op in operations:
        balance -= op
        if balance < 0:
```

Completion

```
            return False
        if balance >= 0:
            return True
    return False
```

With potential bugs

New Benchmarks for Buggy Code Completion

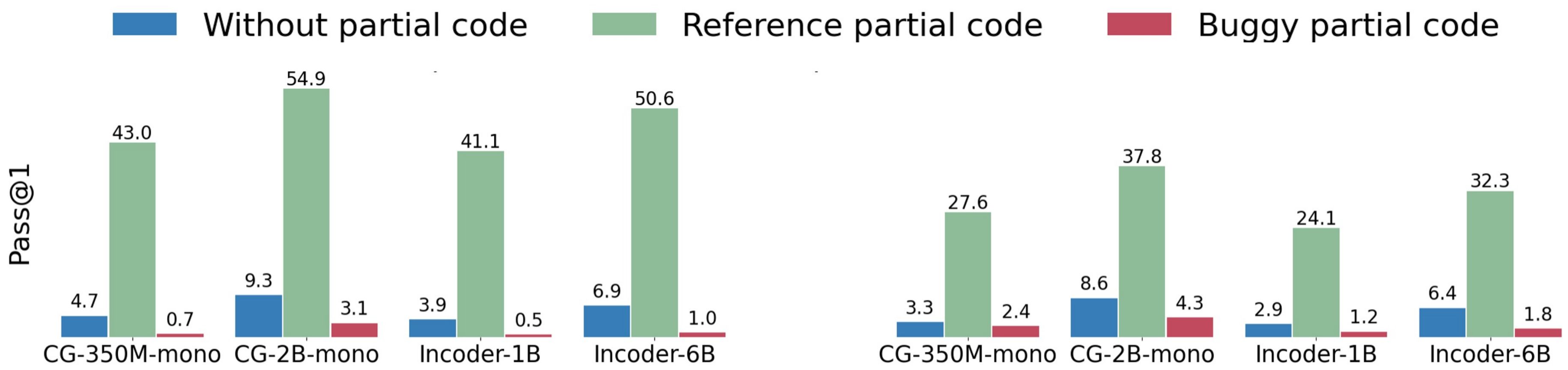
Buggy-HumanEval

- Artificial potential bugs
- Constructed from HumanEval

Buggy-FixEval

- Potential bugs procured from user submissions to coding problems
- Constructed from FixEval and CodeNet

Analysis 1: Failures on Buggy-Code Completion



Performance drops hugely when potential bugs are present!

[CTJ+'21] Chen et al., 2021. Evaluating large language models trained on code.
[HALB'22] Haque et al., 2022. Fixeval: Execution-based evaluation of program fixes for competitive programming problems.

Mitigation Methods for Completion

Strategy 1: **Removal → Completion**

Idea: remove the partial code to guarantee no potential bug exists!

Strategy 2: **Completion → Rewriting**

Idea: treat the completion as buggy and attempt to fix.

Strategy 3: **Rewriting → Completion**

Idea: locate and rewrite potential bugs before being completed

[FAL+'22] Fried et al., 2022. Incoder: A generative model for code infilling and synthesis.

Mitigation Methods for Completion

Strategy 1: **Removal → Completion**

Idea: remove the partial code to guarantee no potential bug exists!

Strategy 2: **Completion → Rewriting**

Idea: treat the completion as buggy and attempt to fix.

Strategy 3: **Rewriting → Completion**

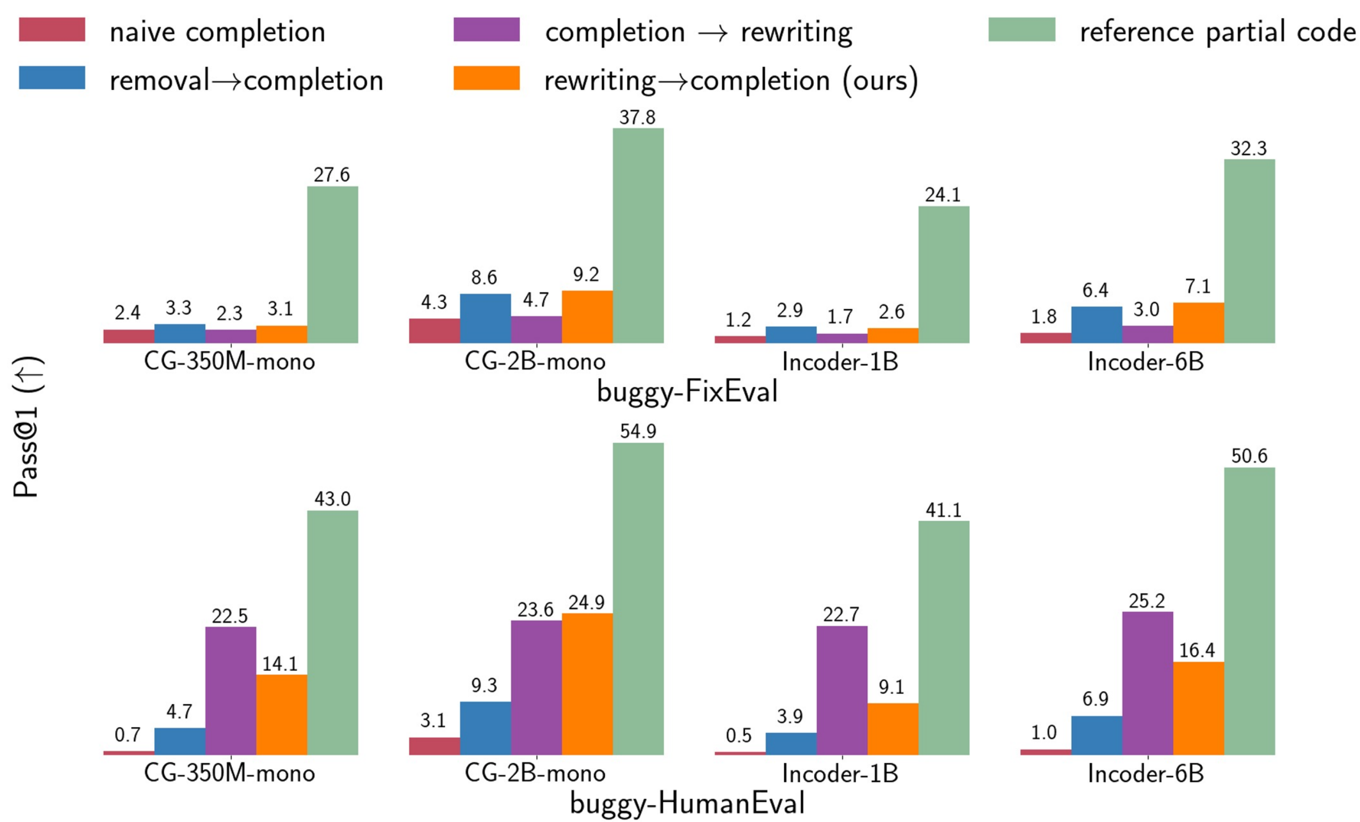
Idea: locate and rewrite potential bugs before being completed

Consider potential bug as distributional outliers
using an infilling model [FAL+'22]

[FAL+'22] Fried et al., 2022. Incoder: A generative model for code infilling and synthesis.

Analysis 2: Mitigation Methods for Completion

Methods improve the completion, but remain huge gap to the inference



Ablation and Case Studies

When do Code-LLMs surpass?

60%: fails to react

```
Check if in given list of numbers, are any two numbers closer
to each other than given threshold.
```

```python
from typing import List
def has_close_elements(numbers: List[float],
                       threshold: float) -> bool:
    for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:

                distance = abs(elem - elem2)
                if distance < threshold:
                    return True
  return False
```

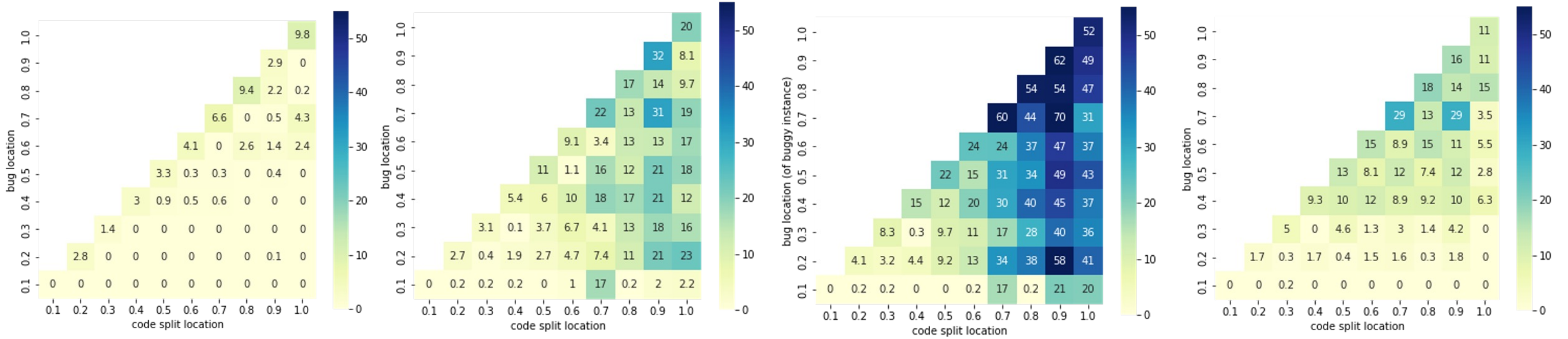
Without potential bugs

```python
from typing import List
def has_close_elements(numbers: List[float],
                       threshold: float) -> bool:
  for idx, elem in enumerate(numbers):
      for idx2, elem2 in enumerate(numbers):
          if idx == idx2:

      continue
          if abs(elem - elem2) < threshold:
              return True
  return False
```

With potential bugs

Bug and split location can affect the performance



Thank you!

Contact information:
Tuan Dinh: tuan.dinh@ucsf.edu
Jinman Zhao: jinmaz@amazon.com

Poster: #539 (Wed 13 Dec 5 -- 7 p.m. CST)
Paper: https://neurips.cc/virtual/2023/poster/70988
Github: https://github.com/amazon-science/buggy-code-completion