

# Enhancing Code Generation via Bidirectional Comment-Level Mutual Grounding

---

Yifeng Di, Tianyi Zhang  
Purdue University, West Lafayette, IN, USA  
di5@purdue.edu, tianyi@purdue.edu

47th International Conference on Software Engineering (ICSE 2025)

# 背景

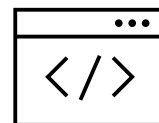
## LLMを用いたコード生成

### - 現状

機能的エラーが多い



要求する機能



生成されたコード

構文エラーはないけど...

期待される出力になっていない

機能的エラーが多いことによって

- ・ 生成されたコードの検査・修正に**労力がかかる**
- ・ 生産性と信頼性が低下

# 既存手法の限界

## 既存手法

- ファインチューニング
- 新しいプロンプト戦略
- 反復的改善手法

などが挙げられるが限定的な改善しか見られない

## 根本原因

- モデルと開発者間での貧弱な双方向コミュニケーション

モデル：開発者の意図を誤解

開発者：コード理解に苦勞

つまり、**認識のズレ**によって期待と一致しないコードが出力される

➡ 認識のズレを解消できればコード生成の精度が上がる

# 目的・アプローチ

---

## 目的

双方向コミュニケーション機構を含んだコード改良手法を提案

## アプローチ

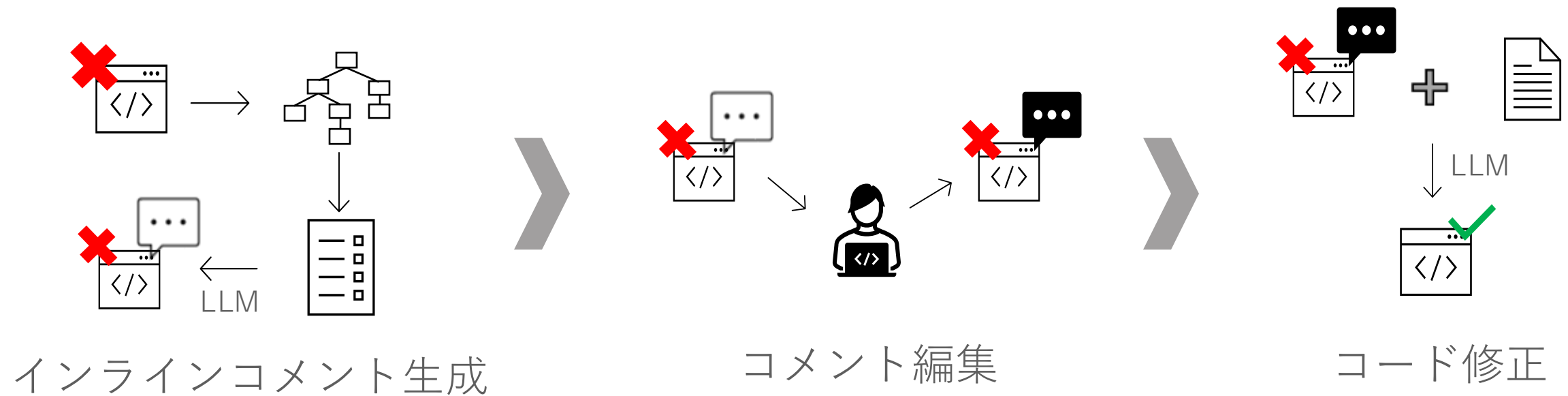
インラインコメントを用いてコミュニケーション機構を実現する

- ・ 評価

複数のLLM・ベンチマークで評価

12名の開発者で有効性を確認

# 提案手法 | 概略



## - インラインコメント生成

コードのASTを解析 → コードセグメンテーション → コメント生成

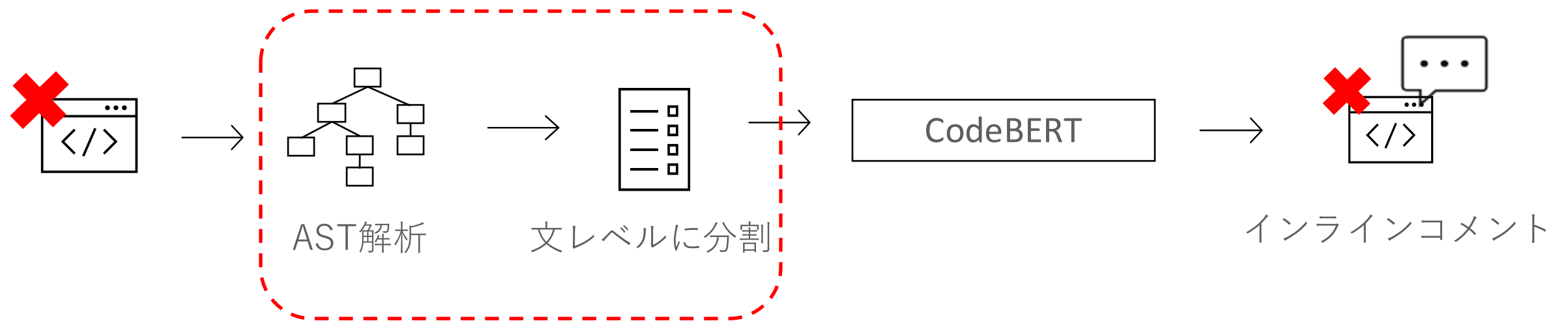
## - コメント編集

人間の手でコメント確認. 機能が間違っているコメントを訂正

## - コード修正

訂正されたコメントと元の問題文をもとにコードを修正

# 提案手法 | インラインコメント生成



## Code Snippet $x$

```
sum_of_digits = 0
```

```
digits = str(N)
```

```
for digit in digits:
```

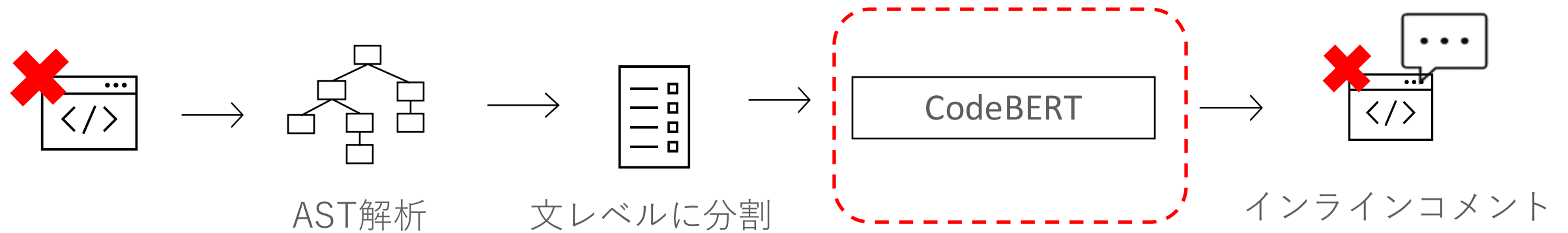
```
    sum_of_digits += int(digit)
```

```
return bin(sum_of_digits)[2:]
```

一文ごとに分割



# 提案手法 | インラインコメント生成



## - CodeBERTのファインチューニング

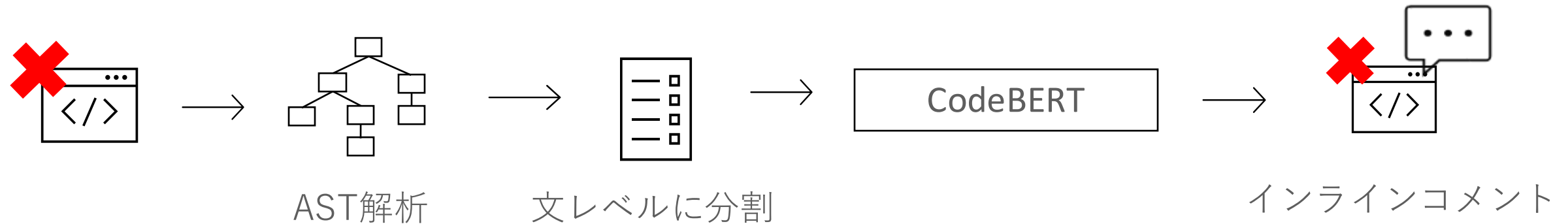
データセット : CodeSearchNet

280,652のコード-コメントペア

入力 : 個別のコード文

出力 : 対応するインラインコメント

# 提案手法 | インラインコメント生成



## Code Snippet $x$

```
sum_of_digits = 0
```

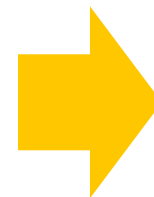
```
digits = str(N)
```

```
for digit in digits:
```

```
    sum_of_digits += int(digit)
```

```
return bin(sum_of_digits)[2:]
```

一文ごとに分割して入力



## 1. Code Snippet with Comment $\hat{x}$

```
# Initialize the variable 'sum_of_digits' to 0  
sum_of_digits = 0
```

```
# Convert N to a string  
digits = str(N)
```

```
# Iterate through each digit in 'digits'
```

```
for digit in digits :
```

```
    # Update 'sum_of_digits' by adding the integer  
    value of 'digit'
```

```
    sum_of_digits += int(digit)
```

```
# Convert the sum of digits to a binary, remove the  
first two characters and return
```

```
return bin(sum_of_digits)[2:]
```

# 提案手法 | コメント修正

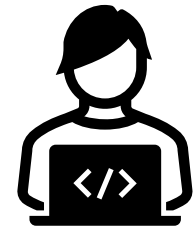
## 1. Code Snippet with Comment $\dot{x}$

```
# Initialize the variable 'sum_of_digits' to 0
sum_of_digits = 0

# Convert N to a string
digits = str(N)
# Iterate through each digit in 'digits'
for digit in digits :
    # Update 'sum_of_digits' by adding the integer
    # value of 'digit'
    sum_of_digits += int(digit)

# Convert the sum of digits to a binary, remove the
# first two characters and return
return bin(sum_of_digits)[2:]
```

ここに機能的エラー

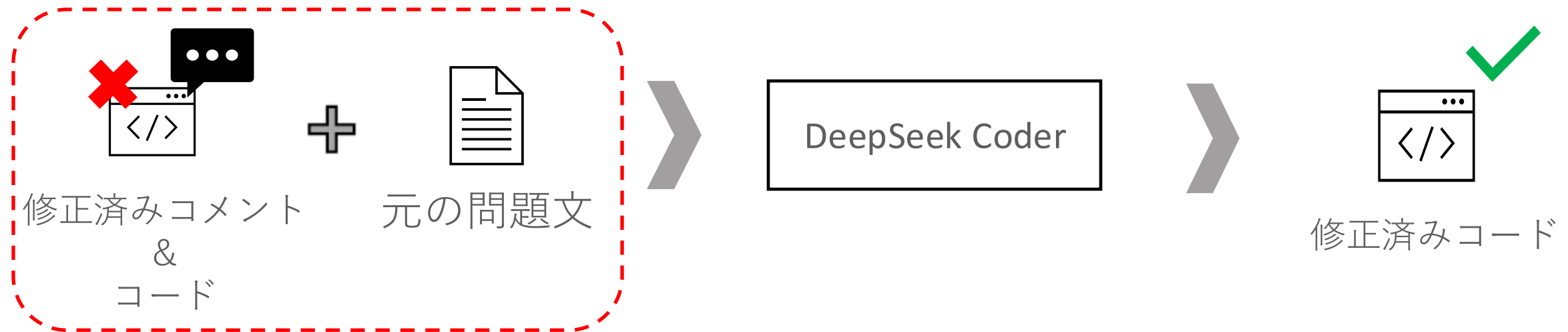


意図した処理になるように  
コメントを変更

## 2. Refine the comment of the incorrect line of code

```
...
- # Convert N to a string
+ # Convert N to a binary, and remove its prefix '0b'
digits = str(N)
...
```

# 提案手法 | コード修正



## 3. Regenerate based on the refined comment

```
# Initialize the variable 'sum_of_digits' to 0  
sum_of_digits = 0
```

変更した部分

```
# Convert N to a binary and remove its prefix '0b'
```

変更したコメントまでのコード

## Context c

```
def solve(N):  
    """Given a positive integer N, return the total  
    sum of its binary digits in binary.  
    Example For N = 1000, the sum of digits will be  
    6 the output should be "110". For N = 150, the  
    sum of digits will be 4 the output should be  
    "100". For N = 147, the sum of digits will be 4  
    the output should be "100".  
    Variables: @N integer Constraints: 0 ≤ N ≤  
    10000.  
    Output: a string of binary number  
    """
```

元の問題文

# 提案手法 | コード修正



## - DeepSeek Coderのファインチューニング

データセット : Stack dataset

190.73GBのPythonコードファイル

フィルタリング

コメント行数の比率（コメント行数/コード行数）が 0.3 ~ 0.95

行うタスク

次トークン予測

# 評価実験

---

## ベンチマーク

- HumanEval 164タスク
- MBPP 142タスク

## 使用モデル（最初のコード生成に利用）

- CodeGen InCoderCode-davinci-002 DeepSeek Coder

合計738の失敗コードを収集

## 修正部分の特定

筆者と大学院生の二人で間違っているコード行を特定・コメント編集

# 実験結果 | RQ1：異なるコード生成モデルでの効果

|                  | HumanEval<br>Pass@1(%) |      |                | HumanEval+<br>Pass@1(%) |      |                |
|------------------|------------------------|------|----------------|-------------------------|------|----------------|
|                  | original               | 提案手法 | 提案手法<br>(FTなし) | original                | 提案手法 | 提案手法<br>(FTなし) |
| InCoder          | 16.5                   | 29.9 | 26.2           | 12.2                    | 25.6 | 23.8           |
| CodeGen          | 29.9                   | 43.9 | 41.5           | 28.0                    | 41.5 | 40.2           |
| Code-davinci-002 | 46.3                   | 63.4 | 63.4           | 42.1                    | 59.1 | 59.1           |
| DeepSeek Coder   | 74.4                   | 79.9 | 78.7           | 70.7                    | 76.2 | 74.4           |

MBPPベンチマークでも同様の結果  
全ての生成モデルで**精度向上**

# 実験結果 | RQ2：既存の生成・修正手法との比較

|                | HumanEval | MBPP   |
|----------------|-----------|--------|
|                | Pass@1    | Pass@1 |
| ReAct          | 56.7      | 66.9   |
| ToT            | 54.3      | 65.5   |
| RAP            | 62.8      | 70.4   |
| Self-Edit      | 62.2      | 56.3   |
| Self-Planning  | 65.2      | 58.5   |
| Self-Debugging | 61.6      | 59.9   |
| SCOT           | 61.0      | 46.5   |
| CodeChain      | 62.8      | 59.2   |
| 提案手法           | 65.9      | 71.1   |

既存手法よりも高精度

# 実験結果 | RQ3：コメント生成モデルによる影響

|                  | Original | Seq2Seq | CodeBERT |
|------------------|----------|---------|----------|
| InCoder          | 16.5     | 23.8    | 26.2     |
| CodeGen          | 29.9     | 39.0    | 41.5     |
| Code-davinci-002 | 46.3     | 60.4    | 63.4     |

比較的単純なモデルを使用しても**2~3%の違い**

人間のフィードバック部分が改善の要因

# 実験 | RQ5：実際のプログラマによる評価

---

## 参加者（12名）

大学院生2名，プロ開発者10名

## 実験タスク(6種類)

- ・ 簡単（2種）

ファイルの読み書き，データ処理・分析

- ・ 普通（2種）

OSとの相互作用，Webページからのデータ抽出

- ・ 難しい（2種）

Webサーバ構築，グラフ・チャートの作成

## 比較対象

GitHub Copilot，Multi-Turn Program Synthesis

# 実験結果 | RQ5：実際のプログラマによる評価

## タスク成功率

|     | 提案手法 | GitHub Copilot | Multi-Turn |
|-----|------|----------------|------------|
| 簡単  | 4/4  | 4/4            | 2/4        |
| 普通  | 4/4  | 3/4            | 1/4        |
| 難しい | 2/4  | 1/4            | 0/4        |

## 信頼度

作ったコードに対する自信（7段階）

提案手法：6.0（中央値）

Copilot：5.0（中央値）

Multi-Turn：2.0（中央値）

Copilotとは有意差なし

Multi-Turnとは有意差あり

# まとめ

---

| LLMと開発者間の認識のズレに対処する方法を提案

| 人間がインラインコメントを編集することでコードを改善

| 従来手法よりも高い精度で生成できた