

Towards Understanding the Characteristics of Code Generation Errors Made by Large Language Models

Zhijie Wang, Zijie Zhou, Da Song, Yuheng Huang, Shengmai
Chen, Lei Ma, Tianyi Zhang

47th International Conference on Software Engineering (ICSE 2025)

背景

LLMを用いたコード生成

```
def change_base(x,base):
```

```
    """
```

実装する機能の説明

```
    """
```



```
def change_base(x,base):
```

```
    if x == 0: return "0"
```

```
    . . .
```

- 自然言語で書かれた機能の説明からコードを生成
- LLMの登場により大きく進歩

課題点

生成されたコードにエラーが含まれる

この研究の着眼点

先行研究のアプローチ

- 高品質訓練データの開発

データ量の拡大・フィルタリングで精度向上

- 指示ファインチューニング技術の改善

人間の指示によりよく従うようにFT (Code Llama など)

- プロンプティング戦略の開発

プロンプトを改善し精度向上

なぜ失敗するかはあまり議論されない



LLMがどのようなケースで失敗するか理解する必要がある

目的・アプローチ

目的

- LLMが生成したコードにはどのようなエラーが含まれるか調査
コード生成を改善する道筋を示す

アプローチ

- HumanEvalデータセットでコード生成しエラーデータを収集
6つの代表的なLLMで生成
(CodeGen-16B, InCoder-1.3B, GPT-3.5, GPT4, SantaCoder, StarCoder)
- 4人の研究者でのオープンコーディングにより質的分析

RQs

RQ

- 1, LLMによって生成されるエラーは異なるか
- 2, 生成されたエラーの修正にはどれくらいの労力が必要か
- 3, タスクの複雑さはコード生成にどのような影響を与えるか

実験方法

エラーデータの収集



LLM	不正解回答数	Pass@1性能
CodeGen-16B	110	32.9%
InCoder-1.3B	114	12.2%
GPT-3.5	42	73.2%
GPT-4	18	89.0%
SantaCoder	139	14.6%
StarCoder	104	34.1%

実験方法

不正確なコード解答の手動分析(RQ1で利用)

- 4人の著者でラベル（エラーの特徴）を作成



意味的特徴（根本原因）と構文的特徴（エラーの場所）を特定

修復労力の分析(RQ2)

- 指標

Levenshtein距離：最小編集回数（挿入・削除・置換の回数）を測定

Jaccard類似度：テキストの重複度

CodeBERTScore：コードの意味的類似度

結果 | RQ 1 (LLM間のエラーの違い)

特定された意味的特徴（一部）

特徴	意味
間違った論理方向	LLMが完全に課題を誤解している
不正確な条件	If文やwhile文の条件式が間違っている
複数ステップの欠落	アルゴリズムの重要な部分が丸ごと欠落
意味のないコードスニペット	課題と全く関係ないコードを生成
条件の欠落	エッジケースや境界条件の考慮不足
定数値エラー	文字列や数値の定数が間違っている

全LLM共通の課題

“間違った論理方向”，“不正確な条件”が全てのLLMで上位

→ 全てのLLMが「課題理解」と「条件処理」で苦戦

結果 | RQ1 (LLM間のエラーの違い)

モデルサイズによる違い

- 大きなモデル (GPT-3.5, GPT-4) の特徴

定数値エラー, 算術演算エラーが多い

無意味なコードを出力することはなかった

- 小さなモデル(上記以外) の特徴

無意味なコードを出力することがある(7~25%)

複数ステップ欠落が多い

全てのエラータイプを示す

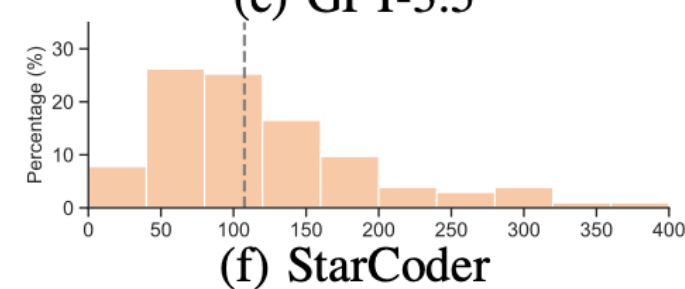
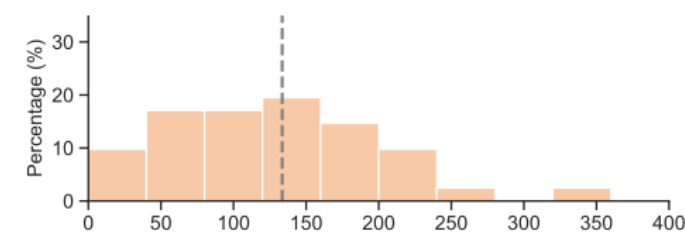
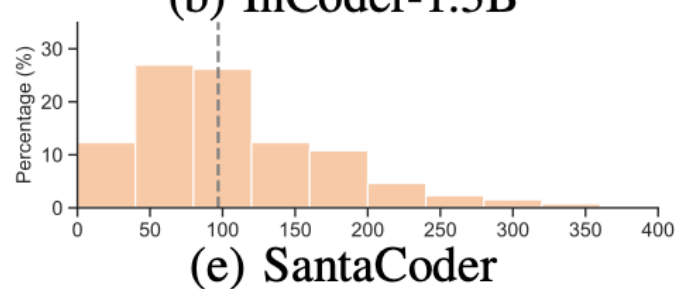
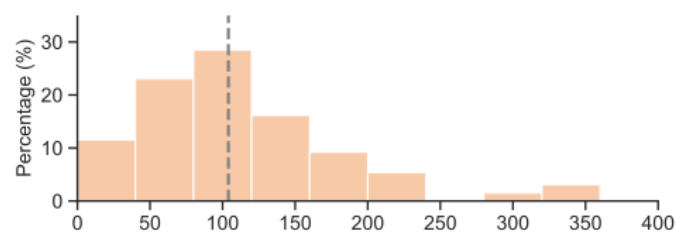
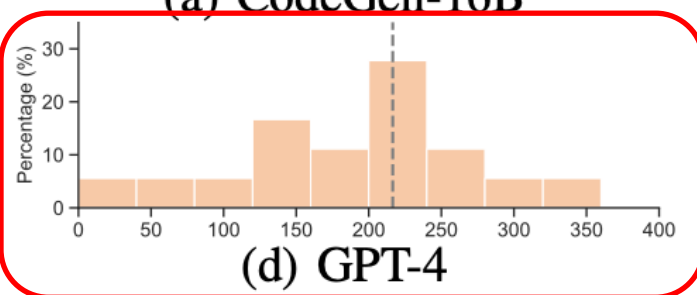
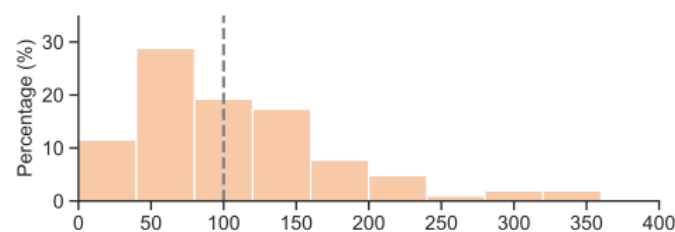
結果 | RQ1 (LLM間のエラーの違い)

RQ1まとめ

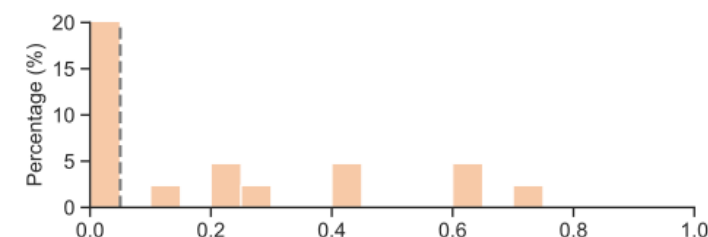
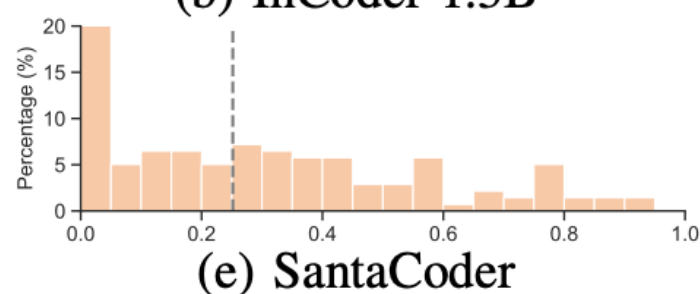
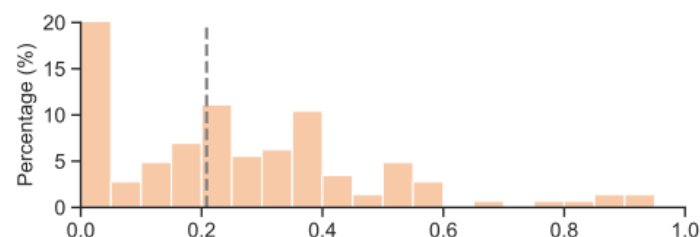
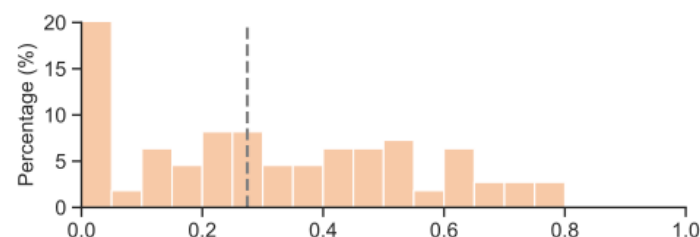
- 1, 全てのLLMが課題理解と条件処理で苦戦
- 2, モデルサイズによってエラーの特徴に差がある
小さなモデルでは無意味なコードを生成しやすい
- 3, 40%以上のエラーが複数行にわたるエラー
→大規模な修正が必要

結果 | RQ 2 (エラー修正の労力)

Levenshtein距離



CodeBERTScore



結果 | RQ 2 (エラー修正の労力)

RQ2まとめ

LLMが生成する不正確なコードの大部分は正解から大幅に逸脱している

- ・ Levenshtein距離

中央値：約100編集以上

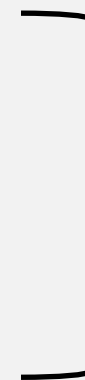
84.21%が50回以上の編集を必要とする

52.63%が200回以上の編集を必要とする

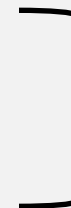
- ・ CodeBERTScore

中央値：約0.2

大部分：0.1以下



→ 大規模な修正が必要



→ 正解とはほぼ無関係

高性能モデルの方がLevenshtein距離が大きくなった

→ 性能と修復容易性には関係がない

結果 | RQ3 (タスク複雑さの影響)

複雑さの定義

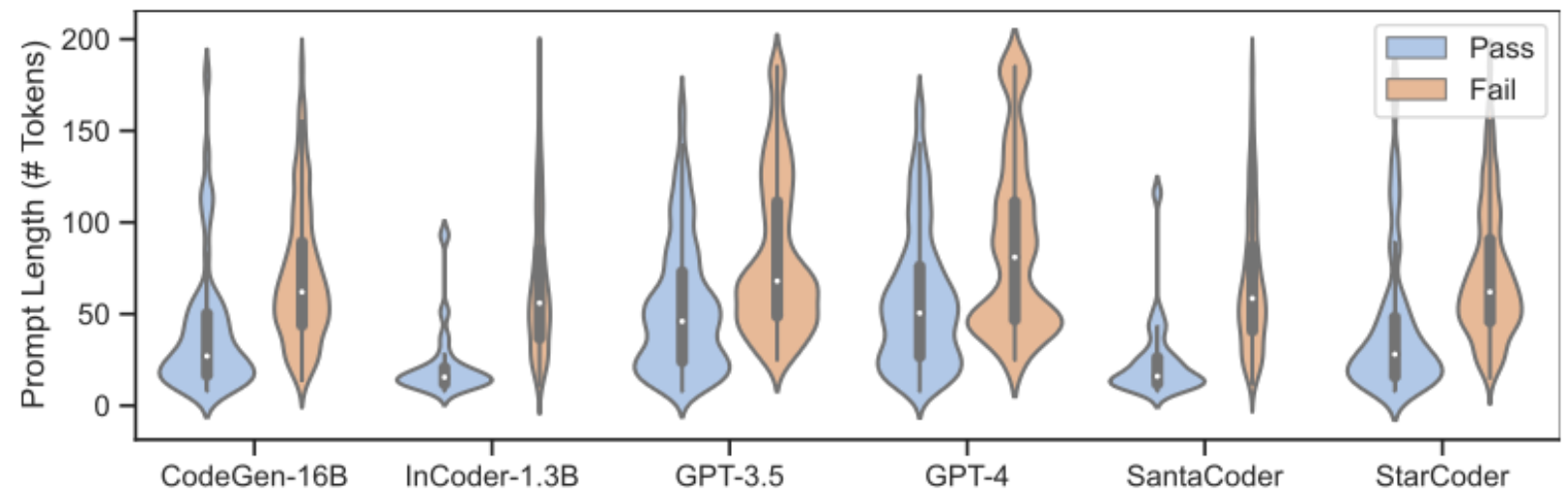
- タスク記述の長さ (プロンプト内の単語数)

成功

中央値 : 50-80語

失敗

中央値 : 80-120語



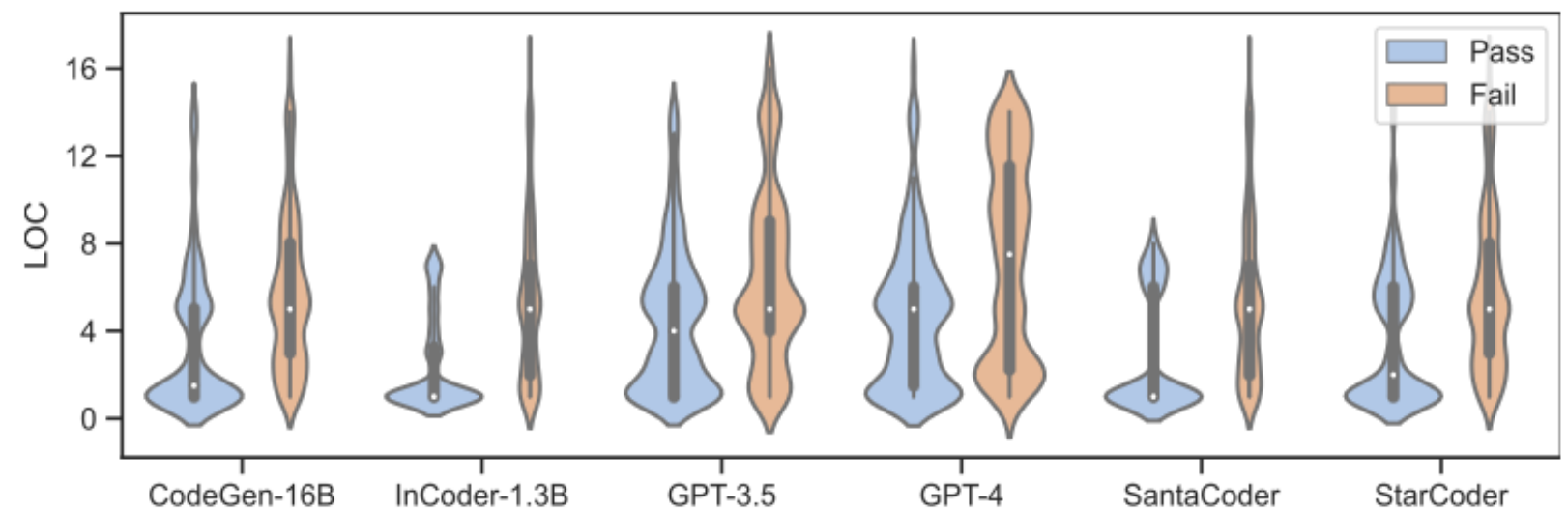
- 正解のコード行数

成功

中央値 : 3-5行

失敗

中央値 : 6-10行



結果 | RQ3 (タスク複雑さの影響)

RQ3まとめ

課題の複雑度と生成結果には明確な**負の相関**がある

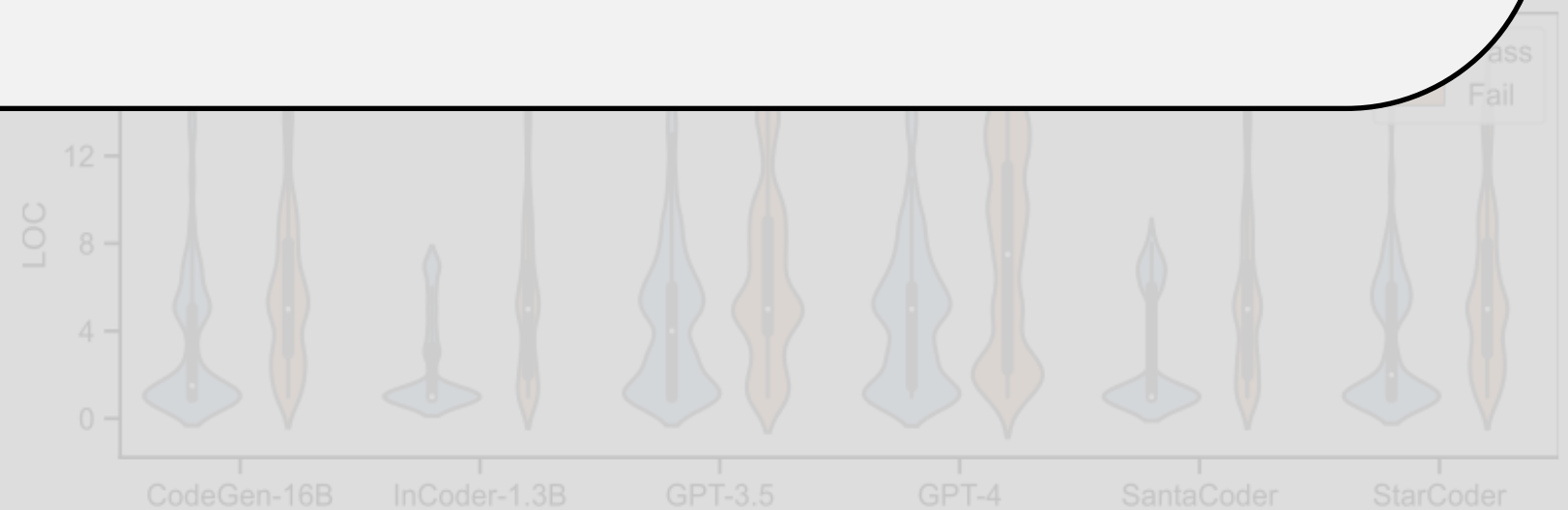
追加の調査により

プロンプトが150語以上 or 正解コードが12行以上だと
要件と関係ないコードを出力して失敗する傾向

中央値：3-5行

失敗

中央値：6-10行



コード生成改善のための提言

自動修復技術の改善

複数行のコードブロックのエラーが多い

→アルゴリズム全体の再設計能力を持つARP技術が必要

生成プロセスの可視化・制御

どのトークンからまちがい始めるのかを調査

生成途中で軌道修正

LLMの能力限界をより精密に解明する

まとめ

| 生成されたコードに含まれるエラーを調査

コード生成の品質改善に寄与

| 課題理解と条件処理で苦戦している

| 不正解の場合は正解と大幅に異なっている

| 課題が複雑になると不正確なコードが出力される傾向

所感

| 最近コードを書かせることが多かったのを読んでみた
みんな薄々感じているような結果だった

| 大きなモデルの方が修復の手間が多いのは意外だった

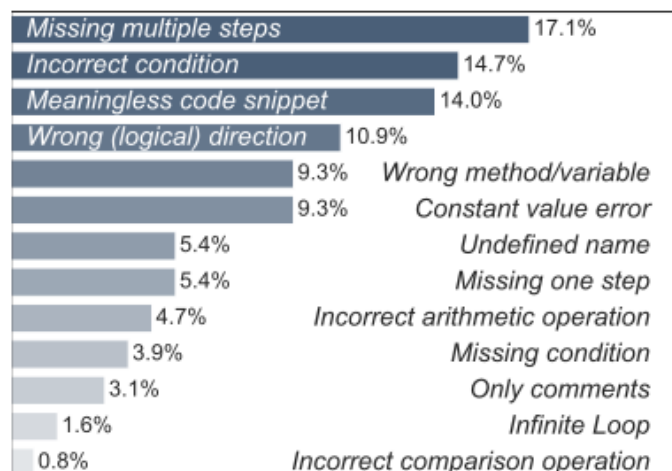
補足資料(意味的特徴)

Error Characteristic		Example of Incorrect Solutions	Correct Solution
Condition Error	Missing condition	<pre># [Task 151] CodeGen-16b def double_the_difference(lst): sum = 0 for i in lst: if i > 0 and i % 2 != 0: sum += i ** 2 return sum</pre>	<pre># [Task 151] Ground Truth def double_the_difference(lst): ans = 0 for num in lst: if num % 2 == 1 and num > 0 and "." not in str(num): ans += num ** 2 return ans</pre>
	Incorrect condition	<pre># [Task 44] GPT-3.5 def change_base(x, base): r = "" while x > 0: rem, r = x % base, str(rem)+r x = x // base return result</pre>	<pre># [Task 44] Ground Truth def change_base(x, base): if x == 0: return "0" ret = "" while x != 0: ret, x = str(x%base) + ret, x // base return ret</pre>
Constant Value Error	Constant value error	<pre># [Task 51] SantaCoder def remove_vowels(text): vowels = 'aeiou' return ''.join([char for char in text if char not in vowels])</pre>	<pre># [Task 51] Ground Truth def remove_vowels(text): return "".join(list(filter(lambda ch: ch not in "aeiouAEIOU", text)))</pre>
Reference Error	Wrong method/variable	<pre># [Task 79] InCoder-1B def decimal_to_binary(decimal): return str(decimal) + "db"</pre>	<pre># [Task 79] Ground Truth def decimal_to_binary(decimal): return bin(decimal) + "db"</pre>
	Undefined name	<pre># [Task 83] CodeGen-16B def starts_one_ends(n): if n == 1: return 1 return starts_one_ends(n-1) + ends_one_starts(n-1)</pre>	<pre># [Task 83] Ground Truth def starts_one_ends(n): if n == 1: return 1 return 18 * 10 ** (n - 2)</pre>
Operation/ Calculation Error	Incorrect arithmetic operation	<pre># [Task 44] InCoder-1B def change_base(x: int, base: int): digits = [] while x: digits.append(str(x % base)) x /= base return ''.join(reversed(digits))</pre>	<pre># [Task 44] Ground Truth def change_base(x: int, base: int): if x == 0: return "0" ret = "" while x != 0: ret = str(x % base) + ret x //= base return ret</pre>
	Incorrect comparison operation	<pre># [Task 138] CodeGen-16B def is_equal_to_sum_even(n): return n <= 8 and n % 2 == 0</pre>	<pre># [Task 138] Ground Truth def is_equal_to_sum_even(n): return n >= 8 and n % 2 == 0</pre>
Garbage Code	Only comments	<pre># [Task 152] InCoder-1B def compare(game, guess): # Regenerate the task description.</pre>	<pre># [Task 152] Ground Truth def compare(game, guess): return [abs(game[i] - guess[i]) for i in range(len(game))]</pre>
	Meaningless code snippet	<pre># [Task 138] StarCoder def is_equal_to_sum_even(n): pass</pre>	<pre># [Task 138] Ground Truth def is_equal_to_sum_even(n): return n >= 8 and n % 2 == 0</pre>
	Wrong (logical) direction	<pre># [Task 20] InCoder-1.3B def find_closest_elements(numbers): closest_to_one, closest_to_two = numbers[0], numbers[1] for number in numbers: if number < closest_to_one: closest_to_one = number if number > closest_to_two: closest_to_two = number return closest_to_one, closest_to_two</pre>	<pre># [Task 20] InCoder-1.3B def find_closest_elements(numbers): min_diff, min_pair = float("inf"), None for l, r in zip(numbers[:-1], numbers[1:]): diff = r - l if diff < min_diff: min_diff = diff min_pair = (l, r) return min_pair</pre>
Incomplete Code/ Missing Steps	Missing one step	<pre># [Task 16] InCoder-1B def count_distinct_chars(string): return len(set(string))</pre>	<pre># [Task 16] Ground Truth def count_distinct_chars(string): return len(set(string.lower()))</pre>
	Missing multiple steps		
Memory Error	Infinite loop	<pre># [Task 100] CodeGen-16b def make_a_pile(n): if n % 2 == 0: return [n] + make_a_pile(n+2) else: return [n] + make_a_pile(n+1)</pre>	<pre># [Task 100] Ground Truth def make_a_pile(n): ans, num = [], n for _ in range(n): ans.append(num) num += 2 return ans</pre>

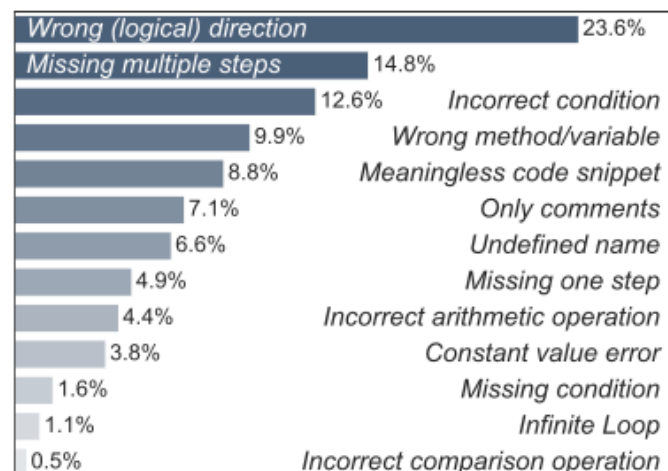
補足資料(構造的特徴)

Error Characteristic		Example of Incorrect Solutions	Correct Solution
Conditional Error	If error	<pre># [Task 151] CodeGen-16b def double_the_difference(lst): sum = 0 for i in lst: if i > 0 and i % 2 != 0: sum += i ** 2 return sum</pre>	<pre># [Task 151] Ground Truth def double_the_difference(lst): ans = 0 for num in lst: if num%2==1 and num>0 and "." not in str(num): ans += num ** 2 return ans</pre>
	For error	<pre># [Task 121] GPT-3.5 def solution(lst): sum = 0 for i in range(1, len(lst), 2): if lst[i] % 2 != 0: sum +=lst[i] return sum</pre>	<pre># [Task 121] Ground Truth def solution(lst): return sum([x for idx, x in enumerate(lst) if idx%2==0 and x %2==1])</pre>
	While error		
Return Error	Incorrect return value	<pre># [Task 103] GPT-3.5 def rounded_avg(n, m) if n > m: return -1 avg=round(sum(range(n,m+1))/(m-n+1)) return bin(avg)[2:]</pre>	<pre># [Task 103] Ground Truth def rounded_avg(n, m) if n > m: return -1 avg = round((n + m) / 2) return bin(avg)</pre>
Method Call Error	Incorrect function name	<pre># [Task 54] StarCoder def same_chars(s0, s1): return sorted(s0) == sorted(s1)</pre>	<pre># [Task 54] Ground Truth def same_chars(s0, s1): return set(s0) == set(s1)</pre>
	Incorrect function arguments		
	Incorrect method call target		
Assignment Error	Incorrect constant	<pre># [Task 138] InCoder-1.3B def is_equal_to_sum_even(n): return n >= 4 and n % 2 == 0</pre>	<pre># [Task 138] Ground Truth def is_equal_to_sum_even(n): return n >= 8 and n % 2 == 0</pre>
	Incorrect arithmetic		
	Incorrect variable name		
	Incorrect comparison		
Import Error	Import error	<pre># [Task 133] StarCoder def sum_squares(lst): return sum[int(math.ceil(i)**2 for i in lst)]</pre>	<pre># [Task 133] Ground Truth def sum_squares(lst): import math return sum(map(lambda x: math.ceil(x)**2, lst))</pre>
Code Block Error	Incorrect code block	<pre># [Task 83] InCoder-1.3B def starts_one_ends(n): count = 0 while n > 0: count, n = count + 1, n / 10 return count</pre>	<pre># [Task 83] Ground Truth def starts_one_ends(n): if n == 1: return 1 return 18 * 10 ** (n - 2)</pre>
	Missing code block	<pre># [Task 60] CodeGen-16B def next_smallest(lst): if len(lst)<2: return None lst.sort() return lst[1]</pre>	<pre># [Task 60] Ground Truth def is_prime(n): if len(lst)<=1: return None sorted_list=sorted(lst) for x in sorted_list: if x!=sorted_list[0]: return x</pre>

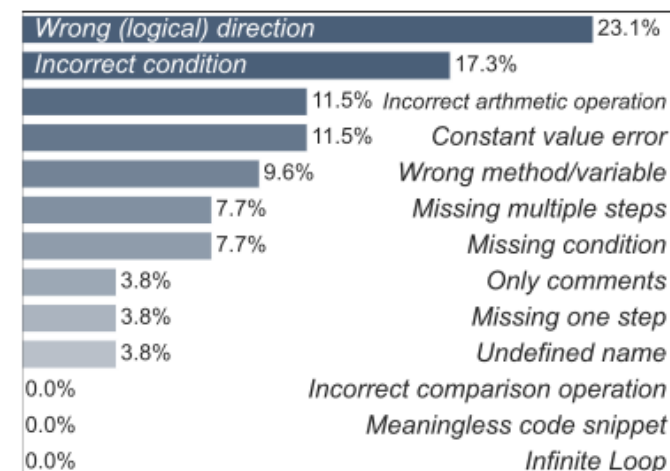
補足資料(RQ1の結果)



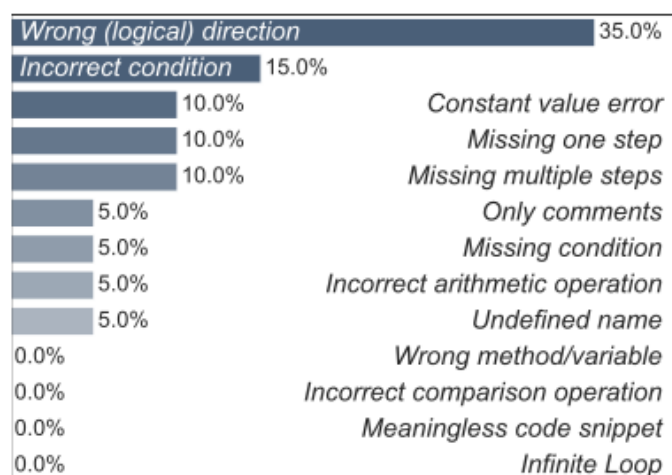
(a) CodeGen-16B



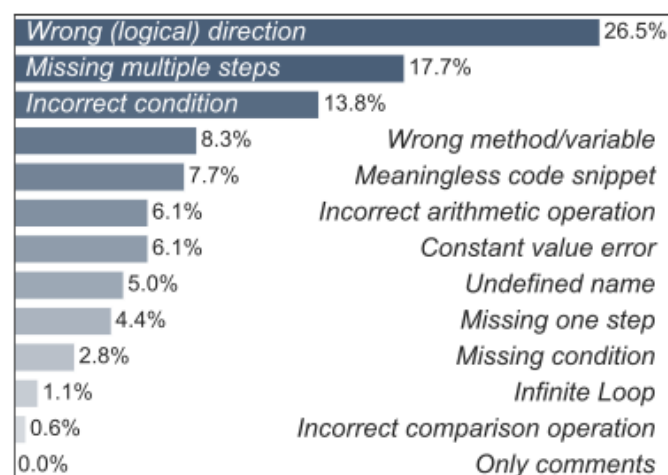
(b) InCoder-1.3B



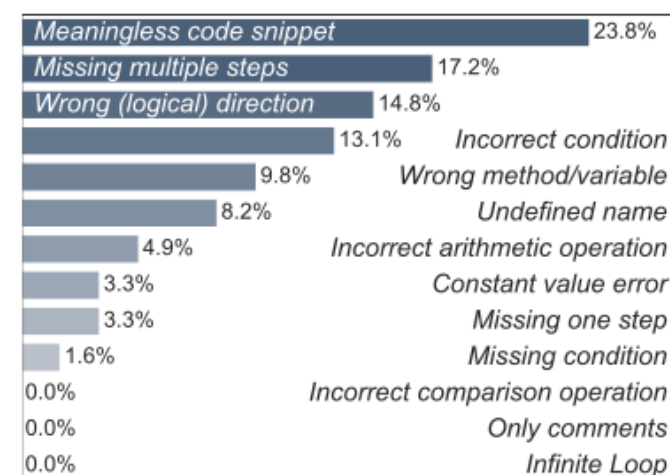
(c) GPT-3.5



(d) GPT-4

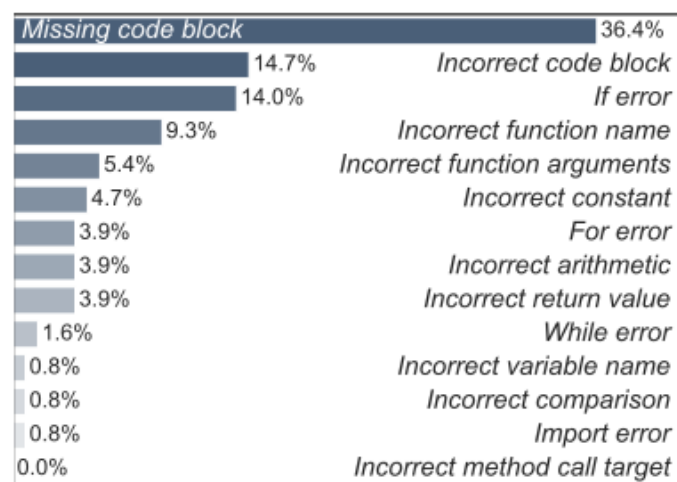


(e) SantaCoder

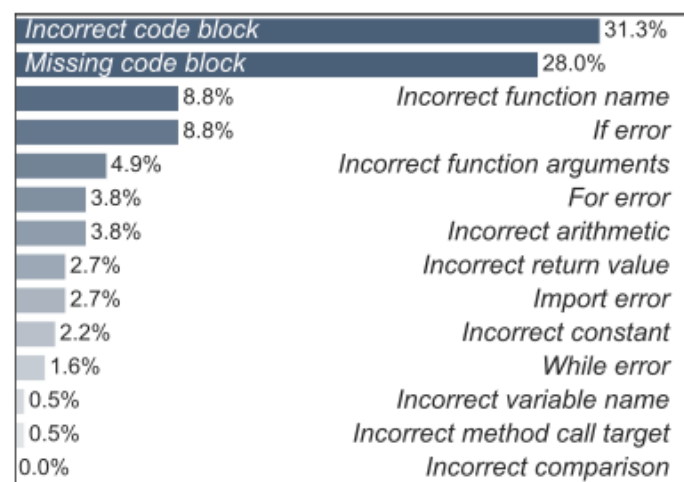


(f) StarCoder

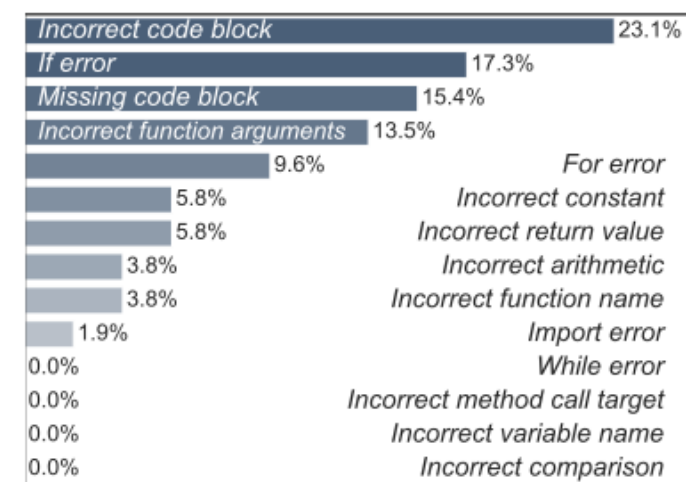
補足資料(RQ1の結果)



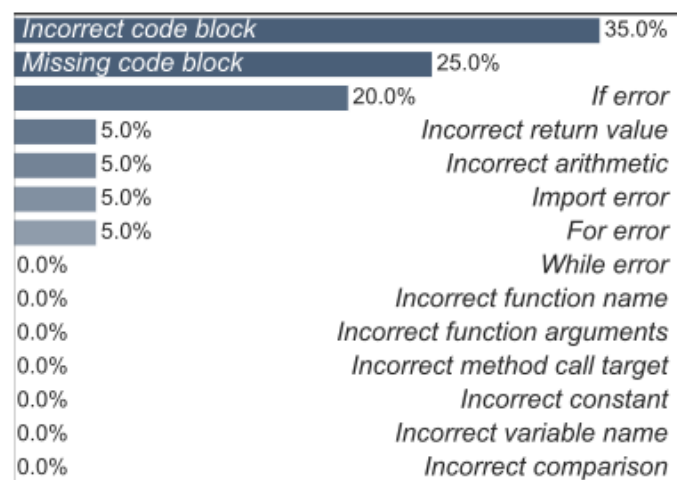
(a) CodeGen-16B



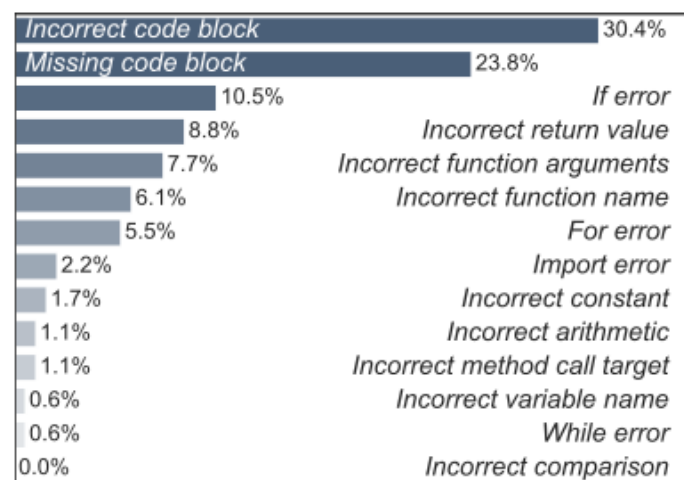
(b) InCoder-1.3B



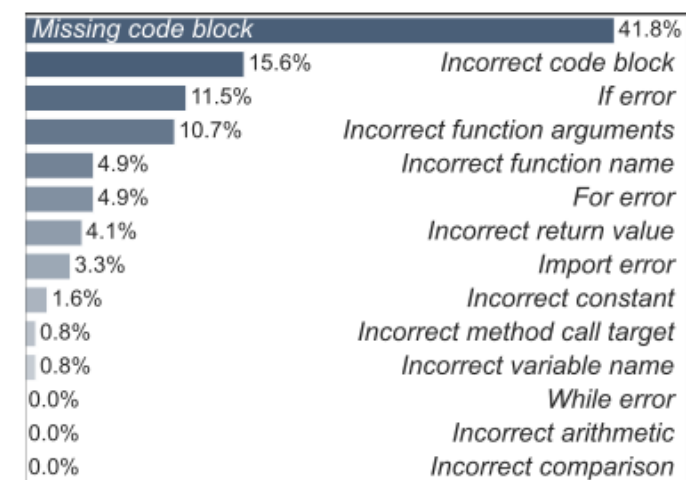
(c) GPT-3.5



(d) GPT-4



(e) SantaCoder



(f) StarCoder