

# A Taxonomy of Inefficiencies in LLM-Generated Python Code

---

Altaf Allah Abbassi<sup>†</sup>, Leuson Da Silva<sup>†</sup>, Amin Nikanjam<sup>‡</sup>, Foutse Khomh<sup>†</sup>

{altaf-allah.abbassi, leuson-mario-pedro.da-silva, foutse.khomh}@polymtl.ca, amin.nikanjam@h-partners.com

<sup>†</sup>Polytechnique Montréal, Montreal, Quebec, Canada

<sup>‡</sup>Huawei Distributed Scheduling and Data Engine Lab, Canada

Proceedings of the 41th International Conference on Software Maintenance and Evolution (ICSME2025)

# コード生成

## LLMを使用したコード生成の研究が盛ん

- 複数のプログラミング言語でコードを生成する能力を発揮<sup>[6]</sup>
  - 様々なタスクで開発者を支援する可能性を示唆<sup>[7]</sup>

## コードの品質上の問題を抱える

- 冗長性, 不必要な計算, 最適とは思えない実装など<sup>[10]</sup>
  - 実行時間の増加, メモリ消費量の増大, 保守性の問題などを引き起こす

機能的に正しくても, 品質上の問題に直面してしまう



開発者にとってソフトウェアの品質に重大なリスクをもたらしてしまう

[6] J. Liu, C. S. Xia, Y. Wang, and L. Zhang, "Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation," Advances in Neural Information Processing Systems, vol. 36, 2024.

[7] T. Sharma, "Llms for code: The potential, prospects, and problems," in 2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C). IEEE, 2024, pp. 373–374.

[10] A. Moradi Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. J. Jiang, "Github copilot ai pair programmer: Asset or liability?" Journal of Systems and Software, vol. 203, p. 111734, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0164121223001292>

# 先行研究・本研究の目的

## LLM生成コードの特定の品質側面を調査<sup>[14,16,17]</sup>

- 正確性, セキュリティ, 保守性など
- より広範な非効率性は大部分が未開拓のまま

## LLMコードにおける非効率正パターンを特定・分類

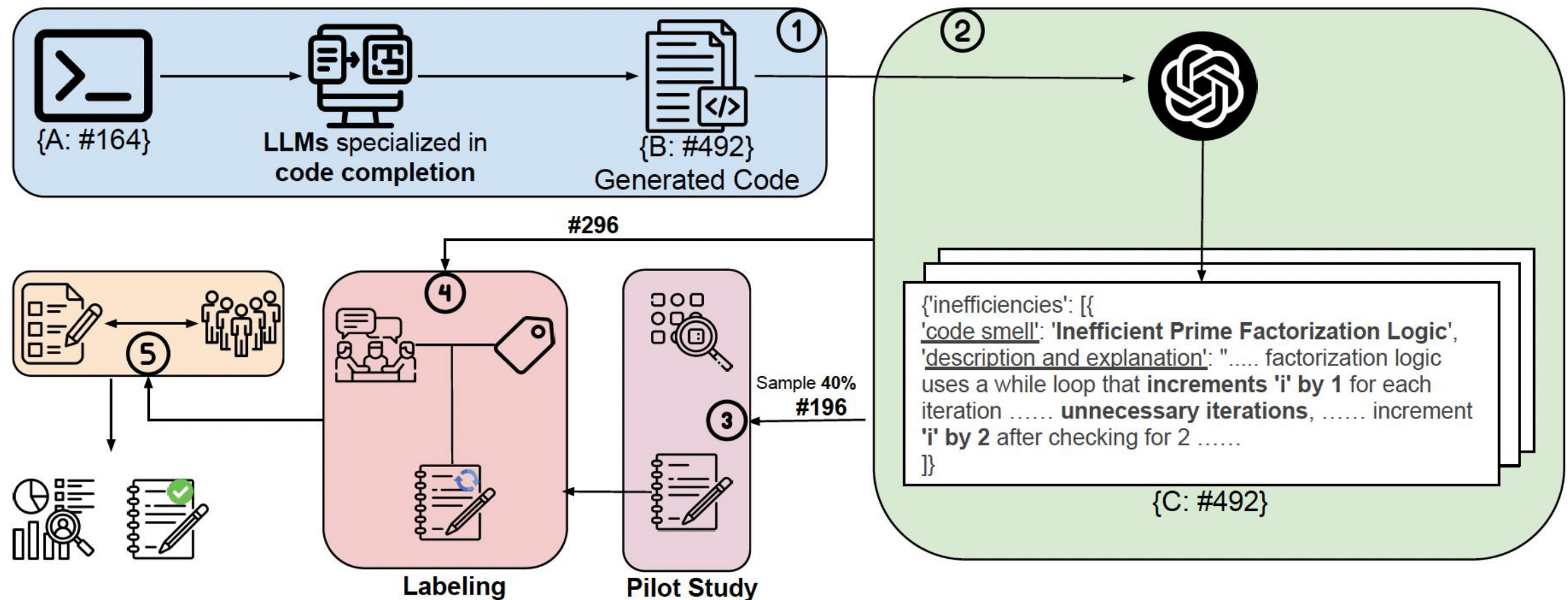
- 非効率性: 品質を低下させ, 大きなシステムの統合を妨げる
- 調査項目
  - RQ1: LLM生成コードにはどのような非効率性パターンが出現するか
  - RQ2: 非効率性は, ソフトウェア研究者にとってどの程度関連があるか

[14] S. Dou, H. Jia, S. Wu, H. Zheng, W. Zhou, M. Wu, M. Chai, J. Fan, C. Huang, Y. Tao et al., “What’s wrong with your code generated by large language models? an extensive study,” arXiv preprint arXiv:2407.06153, 2024.

[16] J. Zheng, B. Cao, Z. Ma, R. Pan, H. Lin, Y. Lu, X. Han, and L. Sun, “Beyond correctness: Benchmarking multi-dimensional code generation for large language models,” arXiv preprint arXiv:2407.11470, 2024.

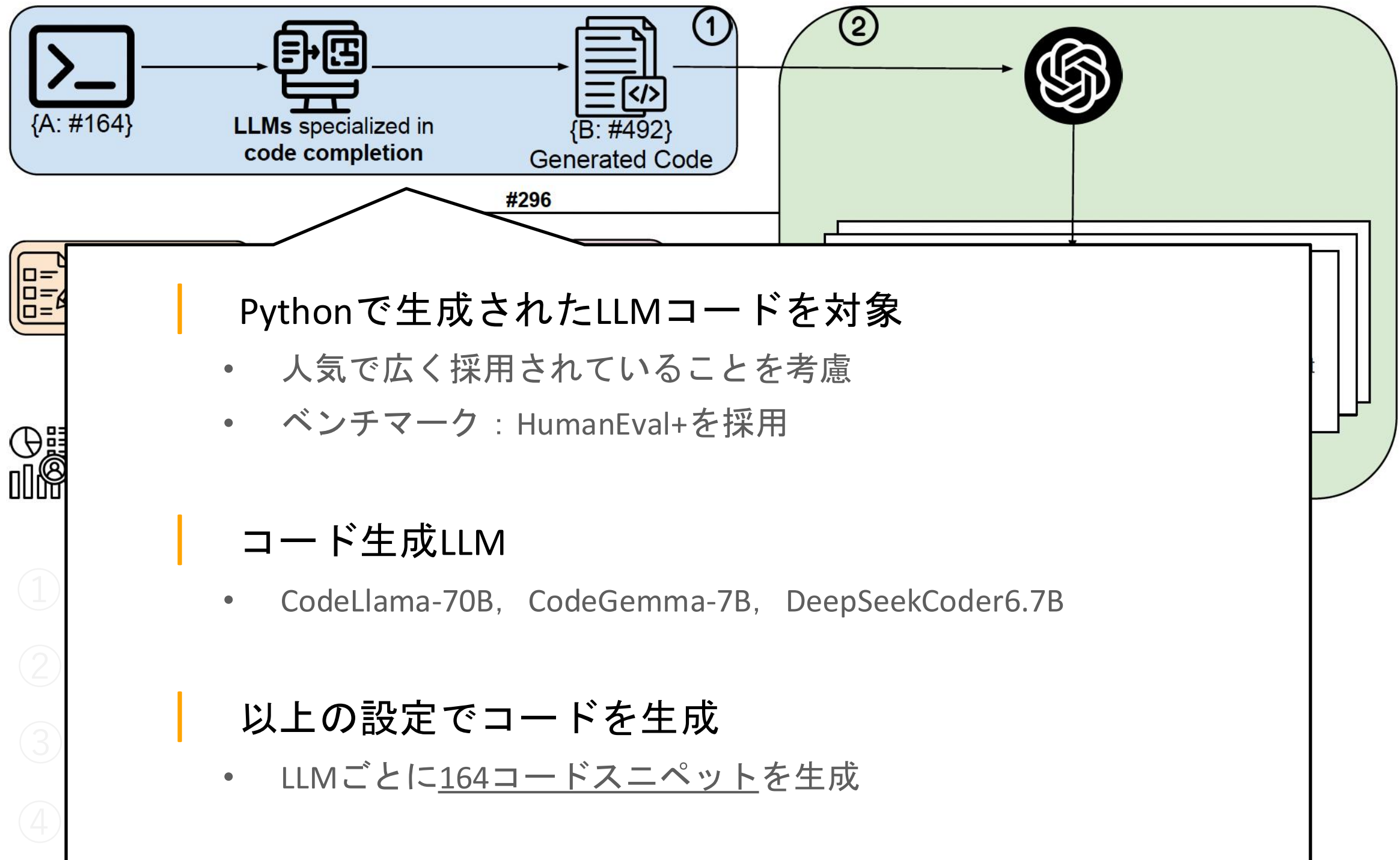
[17] S. A. Licorish, A. Bajpai, C. Arora, F. Wang, and K. Tantithamthavorn, “Comparing human and llm generated code: The jury is still out!” arXiv preprint arXiv:2501.16857, 2025.

# アプローチ：概要



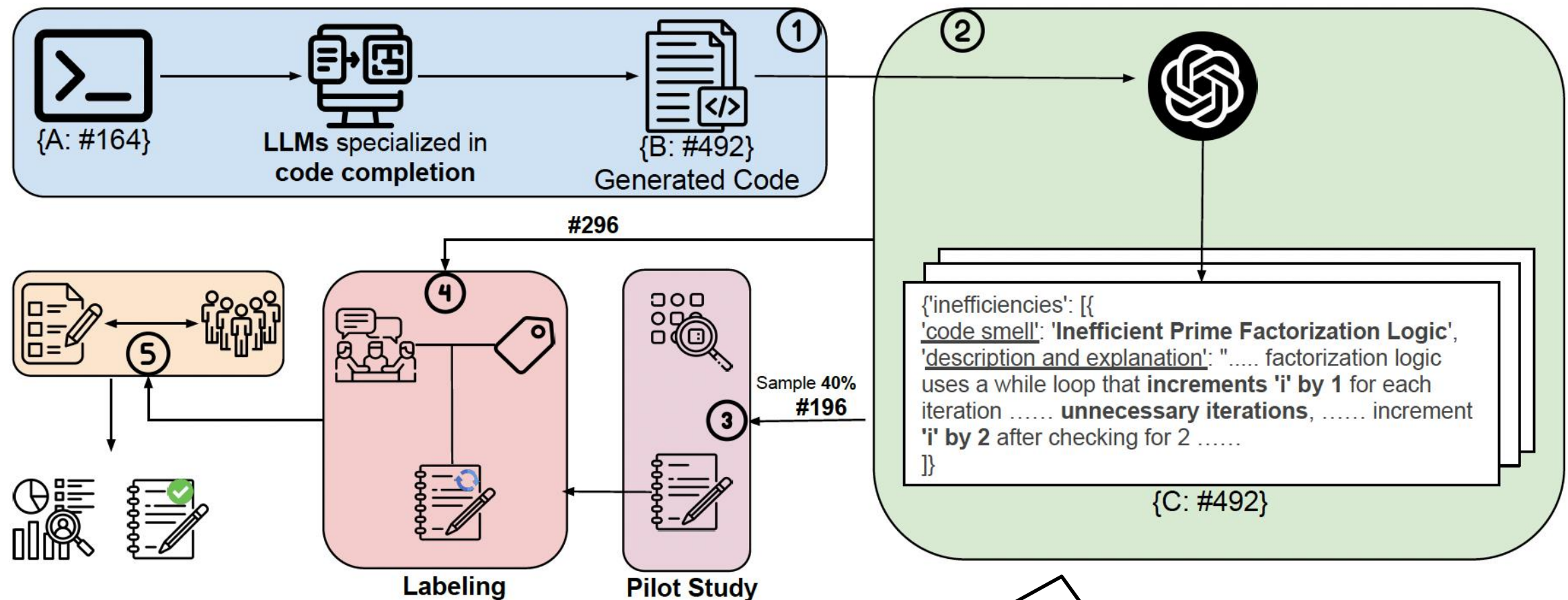
- ①：データの選択と収集
- ②：LLM生成コードの判定
- ③：手動ラベリング，パイロットスタディ
- ④：全データセットのラベリング
- ⑤：参加者のフィードバックとアンケート分析

# ①データ選択・収集



⑤：参加者のフィードバックとアンケート分析

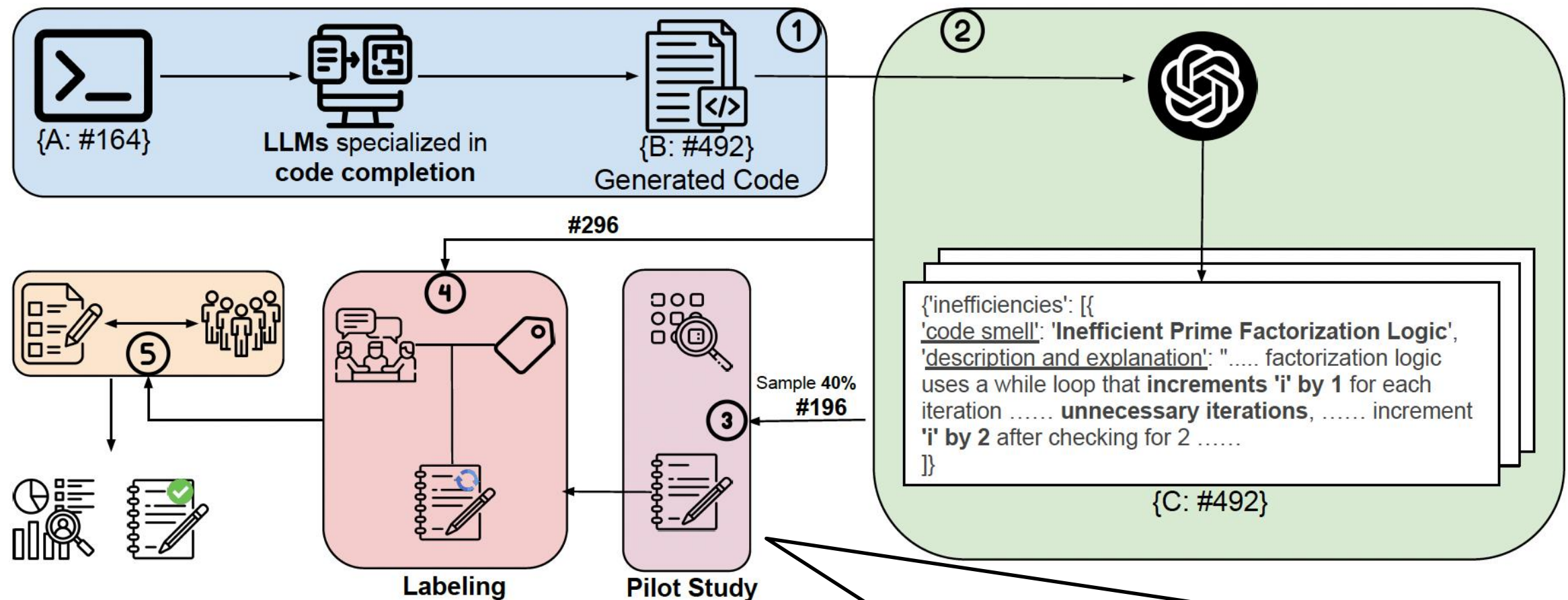
## ② LLM生成コード判定



LLMを活用し，生成コードの品質を評価

- 本研究では，GPT4o-miniモデルを採用
- 生成コードの潜在的な問題点（非効率性）を指摘

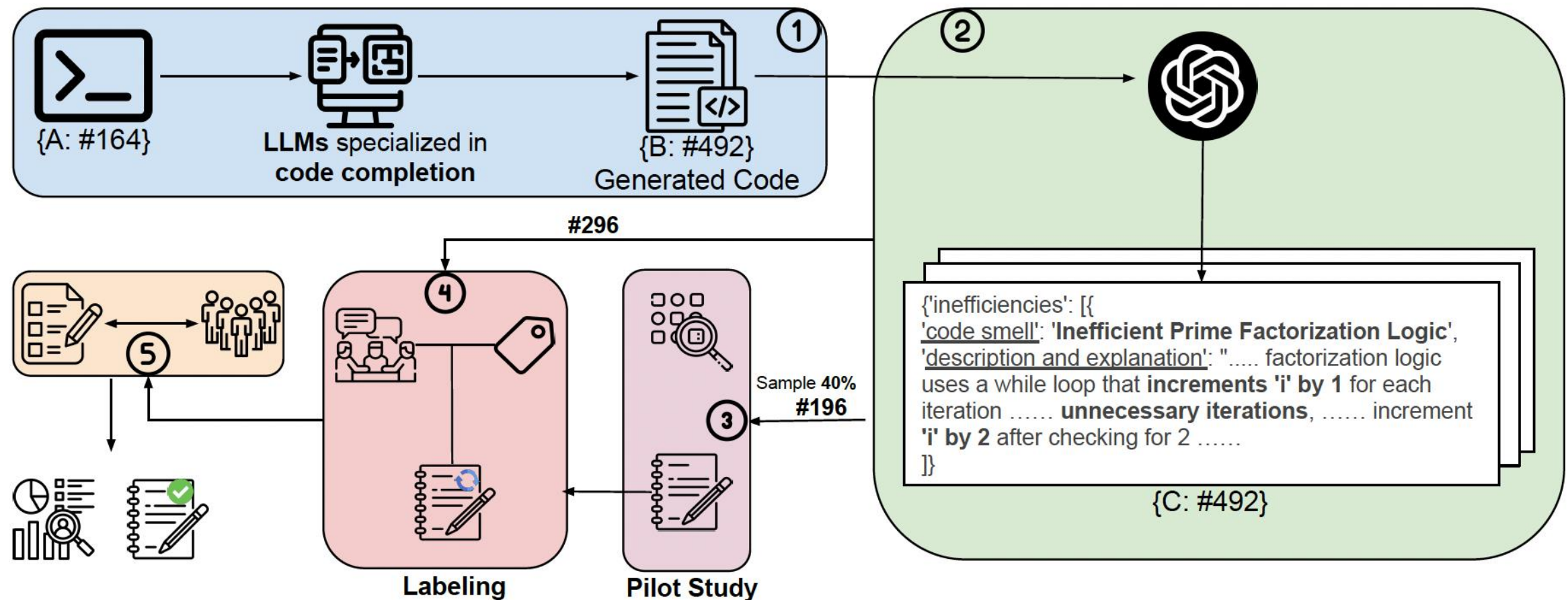
### ③ 手動ラベリング・パイロットスタディ



#### 手動分析を通じ、非効率性のカテゴリーを作成

1. データセットからランダムに196件（40%）抽出し、2名の著者で共同でラベリング
  2. タスク記述、生成コード、コード品質の判定結果を調べ、カテゴリーとサブカテゴリーにグループ化
- 5つのカテゴリーと18のサブカテゴリーからなるパイロットセットを作成

## ④全データセットのラベリング



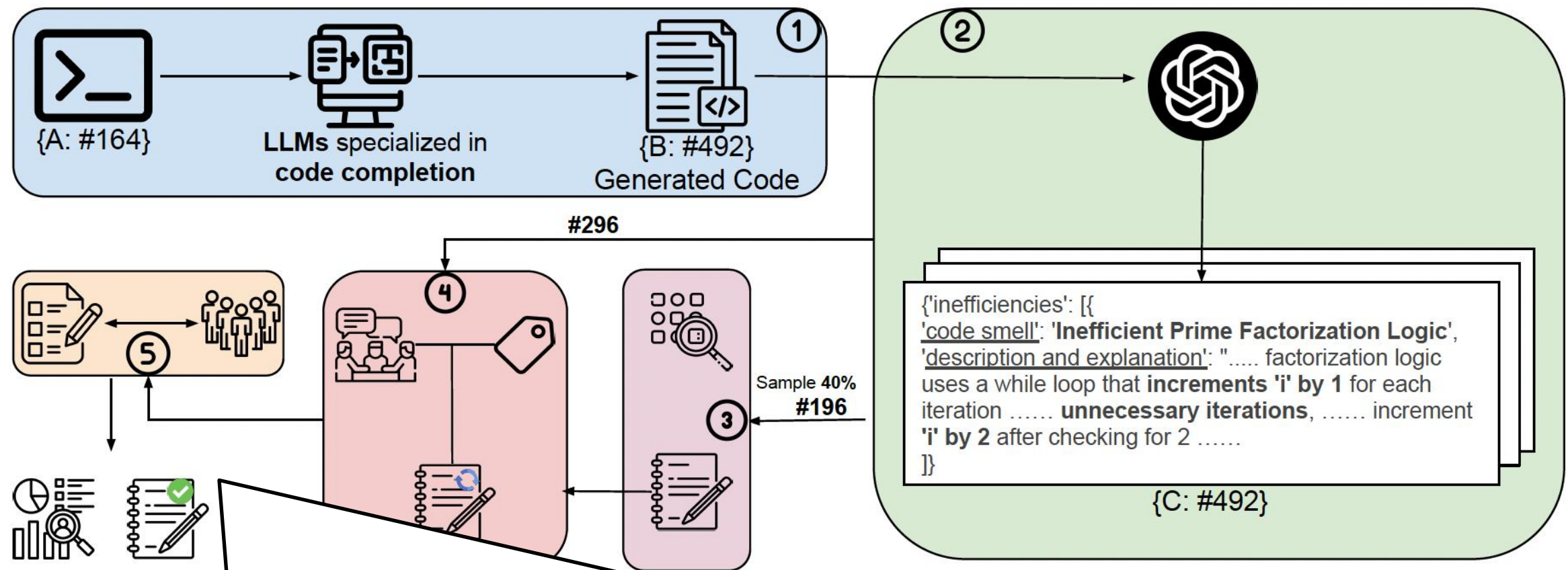
### 残りの60%のサンプルを同様にラベリング

- 分類できないものは「保留」とし、新しいカテゴリーとして必要かを議論
  - 「エラー」のサブカテゴリーを追加
    - 5カテゴリー, 19のサブカテゴリーが完成

### ラベリング終了後, 評価間の一致度を評価

- カッパ係数により算出 (カテゴリー : 0.846, サブカテゴリー : 0.735)

# ⑤ フィードバック・アンケート分析



## コード生成タスクでLLMを使用している研究者を対象

- PyDrillerを使用し, 784件のEメールアドレスを抽出

### アンケート項目

- LLMに関する人口統計学, 自由回答式, 遭遇した非効率性に関する質問
- LLMコードにおける非効率性の頻度と関連性

### ➤ 5段階のリッカード尺度で2つの質問を評価

- 報告された非効率性に遭遇した頻度・対処することがどれほど重要であると信じているか

# RQ1：結果（非効率性のパターン）

---

## 1. 一般的なロジック

- 誤ったロジック

- 生成されたコードに意味のあるロジックがなく，コメントのみが含まれている場合や，メソッド入力をそのまま返している

- 部分的に誤ったロジック

- プロンプトで要求された一部の要件は正しく実装されているが，他の要件が誤って実装されている

- 誤ったメソッド入力

- 不正確，不適切な入力を使用するステートメントを生成している

- 例外・コーナーケース処理

- 生成されたコードに例外処理が欠けている，コーナーケースへの対処に失敗

# RQ1：結果（非効率性のパターン）

---

## 2. パフォーマンス

- 準最適解が存在する（メモリ）
  - 同じロジックをよりメモリ効率の良い方法で実装できるケースが存在
- 準最適解が存在する（時間）
  - LLMで生成されたアルゴリズムと比較して、より時間効率の良いアルゴリズムが存在

ex)

- 時間計算量：より低い時間計算量をもつアルゴリズムに置き換えれる
- 不必要なステップ：出力に影響しないコード内の命令
- 冗長なステップ：ロジックに必要な命令が不必要に繰り返される
- 非効率な反復ブロック：ループの条件が広すぎる

# RQ1：結果（非効率性のパターン）

---

## 3. 可読性

- 紛らわしい変数命令
  - 実行時エラーには繋がらないが、開発者を混乱させる可能性のある変数名
- より読みやすい手法が存在する
  - コードが理解しにくい、またはコーディング規約に準拠していないケース

# RQ1：結果（非効率性のパターン）

---

## 4. 保守性

- コードの重複

- 同一または類似のコードブロックが主要なロジックに貢献せず、複数の場所  
に出現

- コメントの重複

- 反復的、同一、または類似のコメントが過度に使用され、ロジックに関する  
追加の洞察や説明を提供していない

- 条件ブロック

- if ブロックなど、条件ブロックに関する非効率性をグループ化
  1. 不必要なelse：elseが不必要に使用され、削除可能な場合
  2. 不必要な条件ブロック：条件ブロックが不必要な複雑さを加えている

# RQ1：結果（非効率性のパターン）

---

## 5. エラー

- インポートの欠如

- 外部モジュール，ライブラリからのメソッドや定数が適切にインポートされずに使用されている

- 変数宣言の欠如

- 変数が適切に宣言，初期化，またはメソッドの引数として渡されずに使用されている．実行時エラーを引き起こす可能性がある

- 構文エラー

- コード内に構文エラーが含まれており，Pythonにおけるキーワードの不適切な使用や不正確なインデントなどで実行が妨げられる

# RQ1：結果（非効率性の分析）

## 非効率性カテゴリーの頻度

|         | CodeLlama | DeepSeek | CodeGemma |
|---------|-----------|----------|-----------|
| 一般的ロジック | 60.37     | 70.73    | 74.39     |
| パフォーマンス | 48.78     | 30.49    | 23.17     |
| 可読性     | 6.71      | 6.10     | 1.22      |
| 保守性     | 29.88     | 12.8     | 20.73     |
| エラー     | 7.32      | 6.1      | 3.66      |

## 異なるカテゴリー間の共起

- コード内で他のカテゴリーと出現した回数を計測
- 一般的なロジック
  - 可読性(0.74), エラー(0.64), 保守性 (0.58)

欠陥のあるロジックがコードの明瞭さ，構造，長期的な保守性における非効率性にも寄与



# RQ2：フィードバック・アンケート

## 人口統計学情報とLLMを使用したコーディング経験

- 学術界（60%），産業界（40%）を対象
  - 学術界：大学院生（29名），学部生（3名），ポスドク研究員/講師（3名）
  - 産業界：ソフトウェア開発者（11名），データサイエンス開発者（7名），上級役員（4名），サイバーセキュリティ開発者（1名）
- コーディング経験
  - 37名が5年以上，11名が3~5年，10名が1~3年
  - 参加者の98%がコーディングにLLMを使用

## 分類に関する質問の前に2つの課題を議論

- ユーザー体験に関連する非効率性
- コードに関連する非効率性

# RQ2：議論内容

---

## ユーザー体験に関連する非効率性

- Ex（回答人数が多いものを採用）
  - LLMがコードの特定の箇所のデバッグや修正に苦戦
  - LLMが人気のある言語やライブラリに対して、あまり一般的でないものと比較して、より効率的なコードを生成する

## コードに関連する非効率性

- Ex（回答人数が多いものを採用）
  - LLMは単純なタスクもロジックを生成するのに苦勞し、複数回の反復が必要
  - LLMはバージョン追跡に苦勞し、非推奨のメソッドの使用や、ライブラリのバージョンを混在させたり、指定されたAPIと互換性のないコードを生成

# RQ2：非効率性の関連性（アンケート）

## コード非効率性のカテゴリーの関連性(1~5)を計測

- 関連性：非効率性に優先的に対処するべきかどうか

| カテゴリー    | 関連性  |
|----------|------|
| 一般的なロジック | 4.03 |
| パフォーマンス  | 3.52 |
| 可読性      | 3.22 |
| 保守性      | 3.31 |
| エラー      | 3.93 |

- 一般的なロジックが最も関連性が高い
  - 部分的に誤ったロジックが多く，特定の要件を見落とすことが多い  
→ロジックを洗練するために，手動での労力が必要であることを示唆
- パフォーマンスも比較的高い関連性
  - メモリ，時間の最適についての意見が多かった  
→開発者の作業負荷を増やす可能性が示唆

# 総括

---

## 目的

- LLM生成コードの非効率性を特定・分類

## 結果

- 5パターン, 19サブカテゴリーに分類できた
- LLMが効率的なコードを生成する能力に重大なギャップあり
  - 一般的なロジックとパフォーマンスの非効率性が最も優先的に処理されるべきであることが示された