

Are Prompt Engineering and TODO Comments Friends or Foes? An Evaluation on GitHub Copilot

David OBrien
Dept. of Computer Science
Iowa State University
Ames, IA, USA
davidob@iastate.edu

Rabe Abdalkareem
Dept. of Computer Science
Omar Al-Mukhtar University
Elbyda, JK, Libya
rabe.abdalkareem@omu.edu.ly

Sumon Biswas
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
sumonb@cs.cmu.edu

Emad Shihab
Concordia University
Montreal, QC, Canada
emad.shihab@concordia.ca

Sayem Mohammad Imtiaz
Dept. of Computer Science
Iowa State University
Ames, IA, USA
sayem@iastate.edu

Hridesh Rajan
Dept. of Computer Science
Iowa State University
Ames, IA, USA
hridesh@iastate.edu

Proceedings of the IEEE/ACM 46th International Conference on Software Engineering(ICSE '24)

発表者：松尾 和輝

Self-Admitted TD (SATD)

- コードコメントや課題追跡システムで管理される TD[42, 52]
 - “ToDo”といったコメントを用いて管理される

```
def divide(num1, num2):  
    """Divides two numbers"""  
    # TODO: check divide by zero  
    return num1 / num2
```

ToDoコメント

- TDが蓄積するとコード品質や保守性を低下させる
 - 適宜, TDを除去する必要がある

[42] Aniket Potdar and Emad Shihab. 2014. An Exploratory Study on Self-Admitted Technical Debt. In 30th IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, September 29 October 3, 2014. IEEE Computer Society, United States, 91–100. <https://doi.org/10.1109/ICSME.2014.31>

[52] Laerte Xavier, Fabio Ferreira, Rodrigo Brito, and Marco Tulio Valente. 2020. Beyond the Code: Mining Self-Admitted Technical Debt in Issue Tracker Systems. In Proceedings of the 17th International Conference on Mining Software Repositories (Seoul, Republic of Korea) (MSR '20). Association for Computing Machinery, New York, NY, USA, 137–146. <https://doi.org/10.1145/3379597.3387459>

GitHub Copilot

- GitHubが開発したAI支援のコーディングツール
 - OpenAIのCodexモデルを使用しており、数十億行のコードから学習されている
 - 開発者がコードを書く際に自動補完や提案を行い、効率的なコーディングを支援
- ToDoコメントをプロンプトとしてコードを生成することで、TDのないコードを生成できるのでは？
 - 本研究で明らかにする

目的・リサーチクエスチョン

- 目的
 - ToDoコメントが生成されたコードに与える影響の調査
- リサーチクエスチョン
 - RQ1：TODOコメントの有無は，GitHub Copilotが生成するコードの品質に影響するか？
 - RQ2：GitHub Copilotが生成するコードは，TDを除去しているか？
 - RQ3：TODOコメントを修正して，TDを除去する生成コードを導くプロンプトを強化できるか？

データセット (1/2)

- Boaデータセット^[12]内のPythonファイルから,
 - 重複するToDoコメント
 - サードパーティ製コードやライブラリなど, リポジトリ外のToDoコメントを除去した, 29,672件のToDoコメントが対象

Preprocessing Step	TODOs Remaining
Total Extracted	36,381
Unique + Native	29,672
Sampled	380

[12] Robert Dyer, Hoan Anh Nguyen, Hridesh Rajan, and Tien N. Nguyen. 2013. Boa: a language and infrastructure for analyzing ultra-large-scale software repositories. In Proceedings of the 2013 International Conference on Software Engineering (ICSE '13). IEEE Press, San Francisco, CA, USA, 422–431.

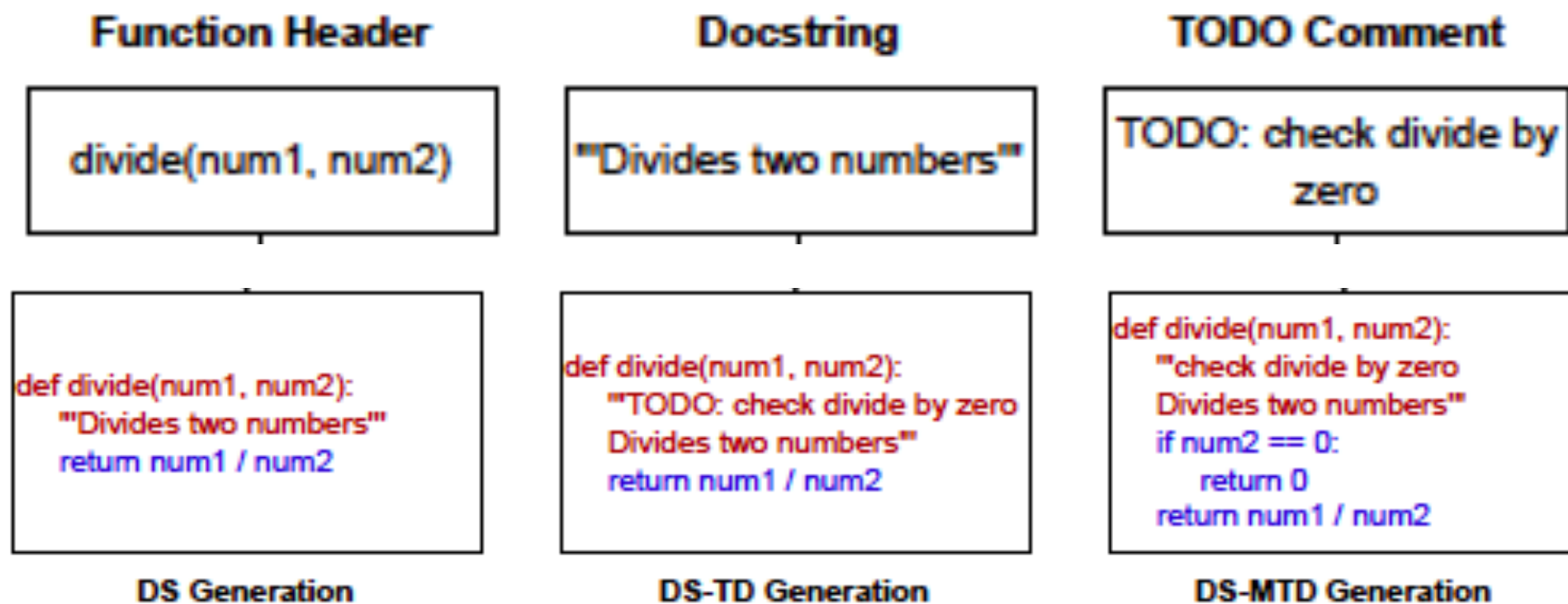
データセット (2/2)

- 29,672件のToDoコメントから、一定の条件でサンプリングする
 - ToDoコメントが関数に関連している
 - ToDoコメントが関数内にある
 - docstringを含む
 - ToDoコメントがあいまいでない など
- サンプリングした380件をデータセットとする

Preprocessing Step	TODOs Remaining
Total Extracted	36,381
Unique + Native	29,672
Sampled	380

プロンプト

- 3種類のプロンプトを定義
 - DS：関数ヘッダーとdocstring
 - DS-TD：関数ヘッダーとdocstringとToDoコメント
 - DS-MTD：関数ヘッダーとdocstringと「ToDo」という単語を除いたToDoコメント
- 3つのプロンプトそれぞれで380件のコードを生成



RQ1 | ToDoコメントが生成に与える影響

- DSとDS-TDで生成したコードでの比較

DSはTDを除去したか

DS-TDはTDを再現したか Does DS-TD Reproduce?	Does DS Repay?		
	No	Yes	Total
No	13	53	66
Yes	285	29	314
Total	298	82	380

- 82件のうち29件（35.36%）で、TDを含むコードが生成されてしまっている
 - ToDoコメントを含まなければ除去できていた

RQ1 | ToDoコメントが生成に与える影響

- 例

- 「ToDo : Document」というToDoコメントが付与
- ドキュメントがないことを示している

```
1 # Load the rule image
2 rule_img = cv2.imread(rule_image_name, cv2.
    IMREAD_GRAYSCALE)
3 # Load the density image
4 density_image = cv2.imread(density_image_name, cv2.
    IMREAD_GRAYSCALE)
5 # Save the density image in temp folder
6 cv2.imwrite('temp/density_image.png', density_img)
7 return rule_img, density_img
```

①DSで生成したコード

```
1 rule_img = cv2.imread(rule_image_name, cv2.
    IMREAD_GRAYSCALE)
2 density_image = cv2.imread(density_image_name, cv2.
    IMREAD_GRAYSCALE)
3 cv2.imwrite('temp/density_image.png', density_img)
4 return rule_img, density_img
```

②DS-TDで生成したコード

①ではドキュメント（コードコメント）が生成されており、TDを除去できている

②ではTDを除去できていない（元のコードを再現）

- Copilotの学習データが原因である可能性が高い

RQ2 | CopilotによるTDの返済

- DS単体での評価

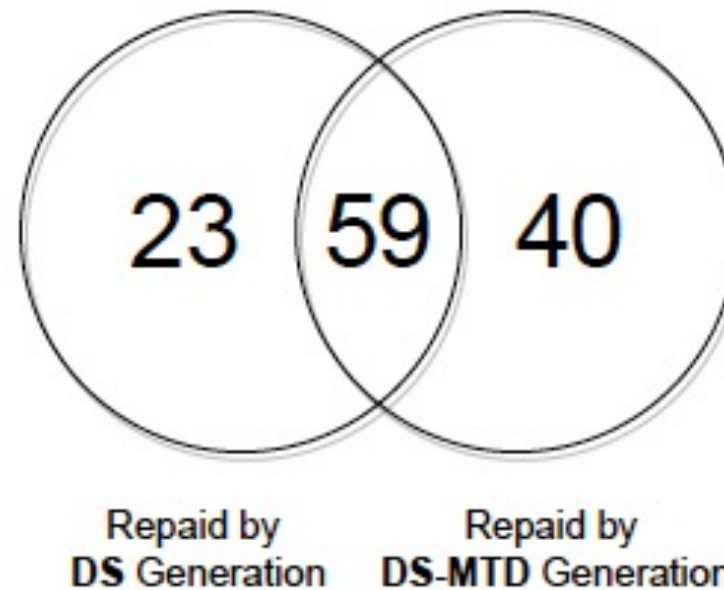
DSはTDを除去したか

DS-TDはTDを再現したか Does DS-TD Reproduce?	Does DS Repay?		
	No	Yes	Total
No	13	53	66
Yes	285	29	314
Total	298	82	380

- 380件のうち82件（21.57%）が除去できている
- 欠陥処理, 追加サポート, 不十分な仮定は除去されやすい

RQ3 | プロンプトエンジニアリングの影響

- DSとDS-MTDを比較



- “ToDo”という単語を削除するだけで、40件のTDを除去することができた
 - 380件のうち122件（32.11%）のTDをcopilotで除去可能
 - プロンプトエンジニアリングは、copilotでのTD除去において重要となる

RQ3 | プロンプトエンジニアリングの影響

- 除去できたTD（上表）とできなかったTD（下表）で、ToDoコメントの特徴分類

名称	説明	プロンプトへの影響	例	割合
具体的な行動	SATDのコメントには、希望する実施または行動を記述することができる。	コード生成は命令に従ってコードを生成するため、これらのSATDコメントはコード生成の目標によく合致している。	todo: support sparse matrix!!, todo: check if the name does not contain forbidden characters:	65.57%
文脈情報	SATDのコメントには、いつ、どこで、どのような変更が行われるのかといった文脈的な詳細が含まれることがある。	適切な文脈を提供するSATDは、コード生成ツールが関連するコードを生成するのに役立つ。	todo: fix code to fail with clear exception when filesize cannot be obtained	32.79%
根拠	SATDのコメントには、返済の理由が記載されている場合がある。	返済の理由を詳細に説明するSATDは、生成のための非機能要件を提供できる。	todo: would it be more efficient using a dict or hash values instead	6.56%
今後の検討事項	SATDのコメントは、変更の検討を意味する場合がある。	DS生成物はこれらの将来の考慮事項を指定されずに行うことがある。	todo: need tau possibly here	27.87%

名称	説明	プロンプトへの影響	例	割合
症状	SATDコメントは具体的な行動ではなく、品質の悪いコードを開示する。	これらのコメントをプロンプトに含めると、生成ツールが解決策ではなく品質の悪いコードを生成する。	todo: untested for glm's?, todo: too much slop permitted here impossible, todo# too long?	14.34%
近接性	SATDコメントは元のコード内の近くを参照する。	元のコードにアクセスできないため、これらのコメントと特定のコードセグメントとの関係が失われ、コード生成のパフォーマンスが妨げられる。	todo: fix next line, todo: clean this up, todo: complete this documentation	32.95%
質問	SATDコメントはコードの品質に疑問を投げかける。	プロンプトに挿入されると、解決策ではなく疑問のある品質のコードが生成される。	todo: remove redundant attributes and fix the code that uses them?, todo: how to accommodate regression?	15.12%

まとめ

- 目的
 - ToDoコメントが生成されたコードに与える影響の調査
- 結果
 - ToDoコメントを用いることで、一部のTDを除去可能
 - うまくプロンプトエンジニアリングを行うことで、より多くのTDを除去できる可能性を示唆
- 所感
 - ToDoという単語を削除しただけで、なんでTDを除去できるのかがよくわからん
 - 生成AIの発展が早すぎて、こういうプロンプトエンジニアリング関連の研究はすぐに過去のものになりそう（もうなってる？）