Analyzing the Impact of Self-Admitted Technical Debt on the Code Completion Performance of Large Language Models

1. Background

Self-Admitted Technical Debt (SATD) refers to suboptimal code introduced during rushed development, often as quick and temporary fixes, which developers explicitly acknowledge through source code comments. Common examples include comments like `TODO` or `FIXME`:

```
// TODO: handle null case
```

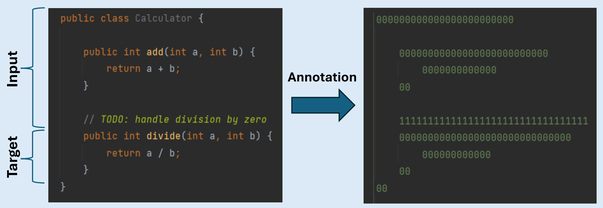
Removing SATD from training data has shown to improve model performance. However, the isolated effect of SATD at inference time remains unclear.

2. Research Questions

1. What is the presence of SATD in The Heap?
2. What is the impact of SATD on the performance of LLMs for code completion tasks?
3. Do models generate correct code, that doesn't match (broken) ground truth?

3. Methodology

1. SATD Detection + Annotation
2. Input–Target Construction
3. Code Generation
4. Evaluation: Quantitative + Qualitative




```
Models = SmolLM2, StarCoder2, or Mellum
Prediction = model.generate(input)
Score = compare(prediction, target)
```

4. Results

RQ1 Presence of SATD in the Heap

Table 1: SATD presence statistics in the Java subset of The Heap

Measure	Value
Total source files	5,168,193
Total SATD comments	914,347
Files with ≥1 SATD comment	431,145 (8.34%)
Average SATD comments per file	0.18
Median SATD comments per file	0
Max SATD comments in a single file	2,230
SATD comments per KLOC	1.26

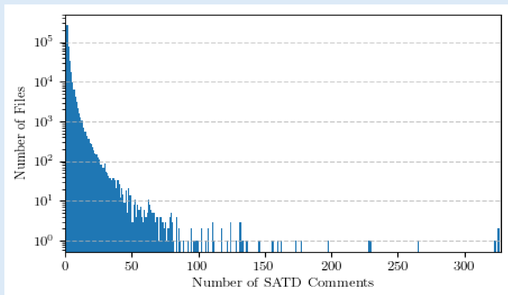


Figure 1: Distribution of SATD comments per file in the The Heap Java dataset

RQ2 Quantitative Model Performance

Only the BLEU metric is presented here since all other metrics shared the same trends

- **No-smell-comment performs better** than in-smell
- Methods **farther from SATD** perform **better**
- **Peprocessing** by removing SATD from the context has **negligible effects**
- **Bigger** models perform **better**

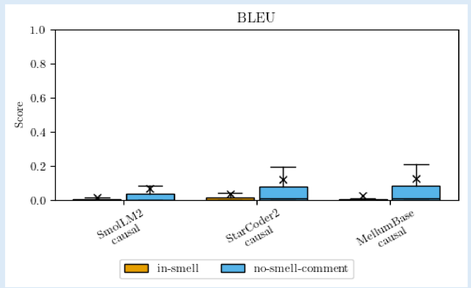


Figure 2: Metric scores for group 1

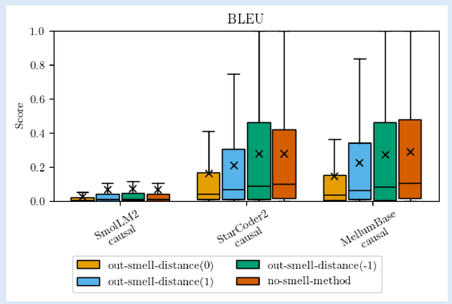


Figure 3: Metric scores for group 2

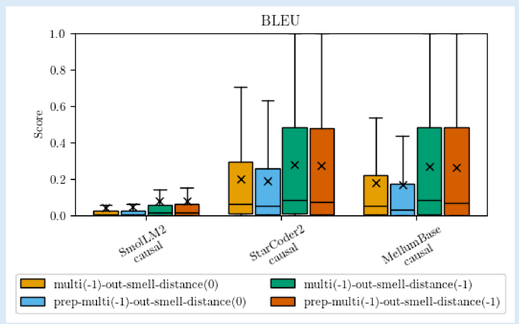


Figure 4: Metric scores for group 3

RQ3 Qualitative Model Performance

- **Trends** in semantic classification **largely matched** the text-based metrics
- **Shorter targets** achieved **higher metric scores**, often due to common boilerplate (e.g., public void).
- **Preprocessing SATD** had little impact, models often produced **nearly identical outputs.**
- In only **6 of 40 files** was SATD located directly above the target method. **Two** of those had debt that could be fixed. The models still **failed to fix** the described technical debt.

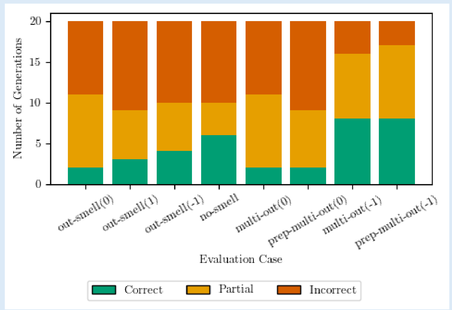


Figure 5: Qualitative evaluation of semantic generation correctness (causal only)

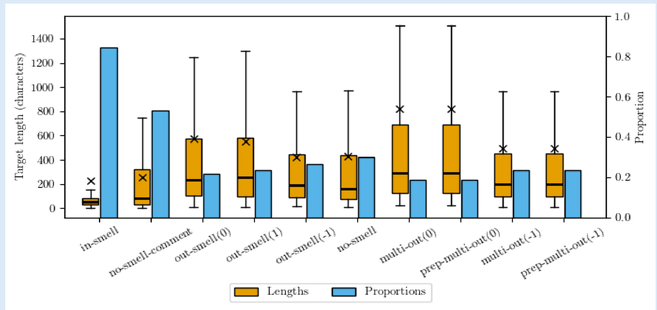


Figure 6: Lengths (in characters) and proportions of short methods (three lines or fewer) per case

5. Conclusion

- SATD is sparse in the dataset
- SATD has a negligible impact on performance
- Three other factors had more impact on performance:
  - Target length
  - Method complexity
  - Available context
- Metric scores generally aligned with semantic correctness
- Metrics are misled by non-functional text
- No evidence that models were able to generate correct completions in cases where the ground truth was broken.

6. Future Work

- Try larger models
- Rerun FiM generations
- Try different languages
- Explore other data smells
- Try more localized SATD targets

SCAN ME

Author: Lucas Witte (L.C.Witte@student.tudelft.nl) **Supervisors**: Jonathan Katzy and Razvan Mihai Popescu **Responsible Professors**: Maliheh Izadi and Arie van Deursen

TU Delft