Large Language Models for Code Refactoring

PhD Thesis Project

Researcher: Henrique Nunes

Advisor & Coadvisor: Eduardo Figueiredo & Tushar Sharma

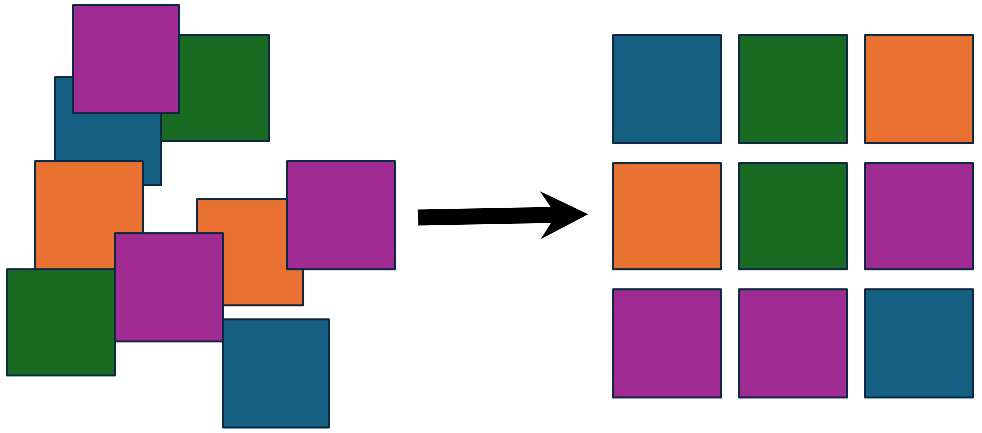
Context of Our Work

Why LLMs for Code Refactoring?

lab·soft
software engineering laboratory

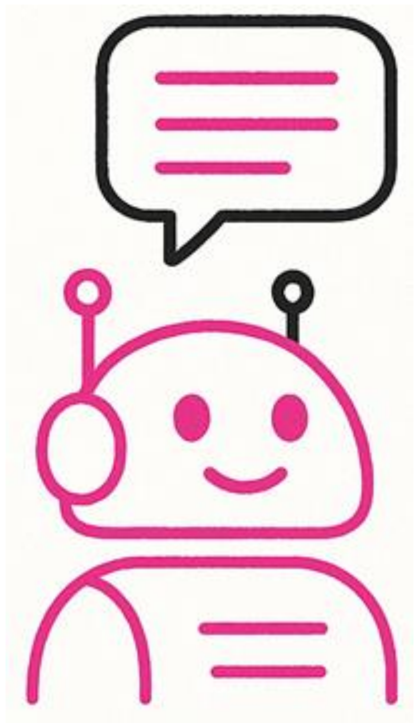
Refactoring

- Refactoring reduces code **complexity** and improves **maintainability**.
- Manual refactoring is costly and slow.
- Refactoring tools are semi-automated, inaccurate and require human intervention.



Large Language Models (LLMs)

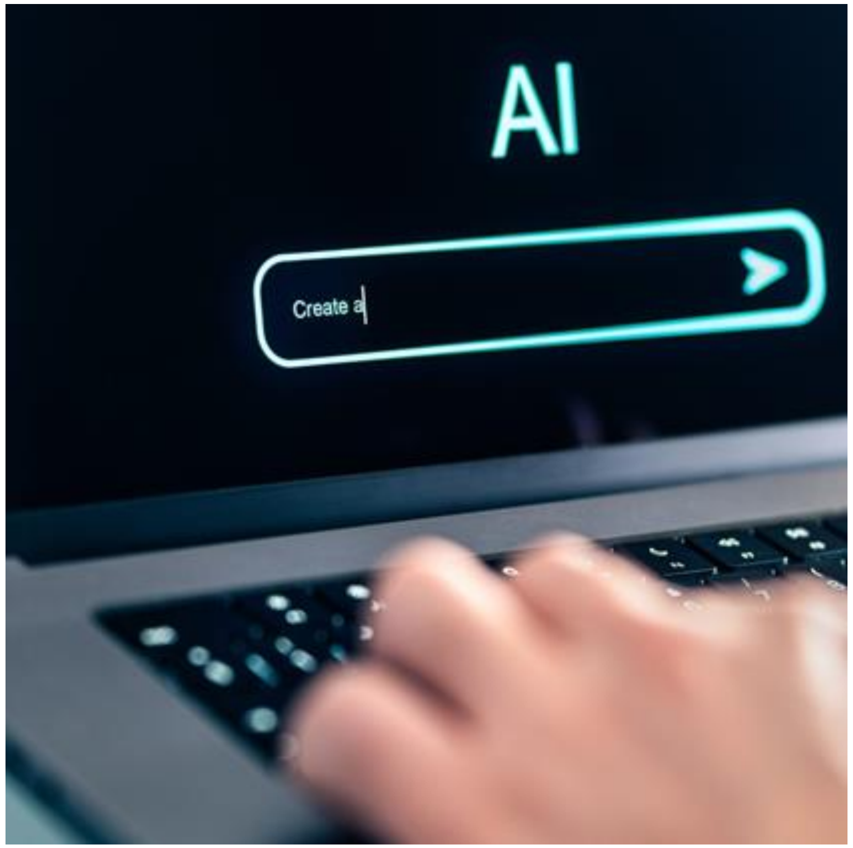
- LLMs are models trained on a large amount of data and have shown good capability to **write and understand natural language**.
- Studies on LLMs are growing exponentially in software engineering, yet their capability show **limitations** and still need to be explored.



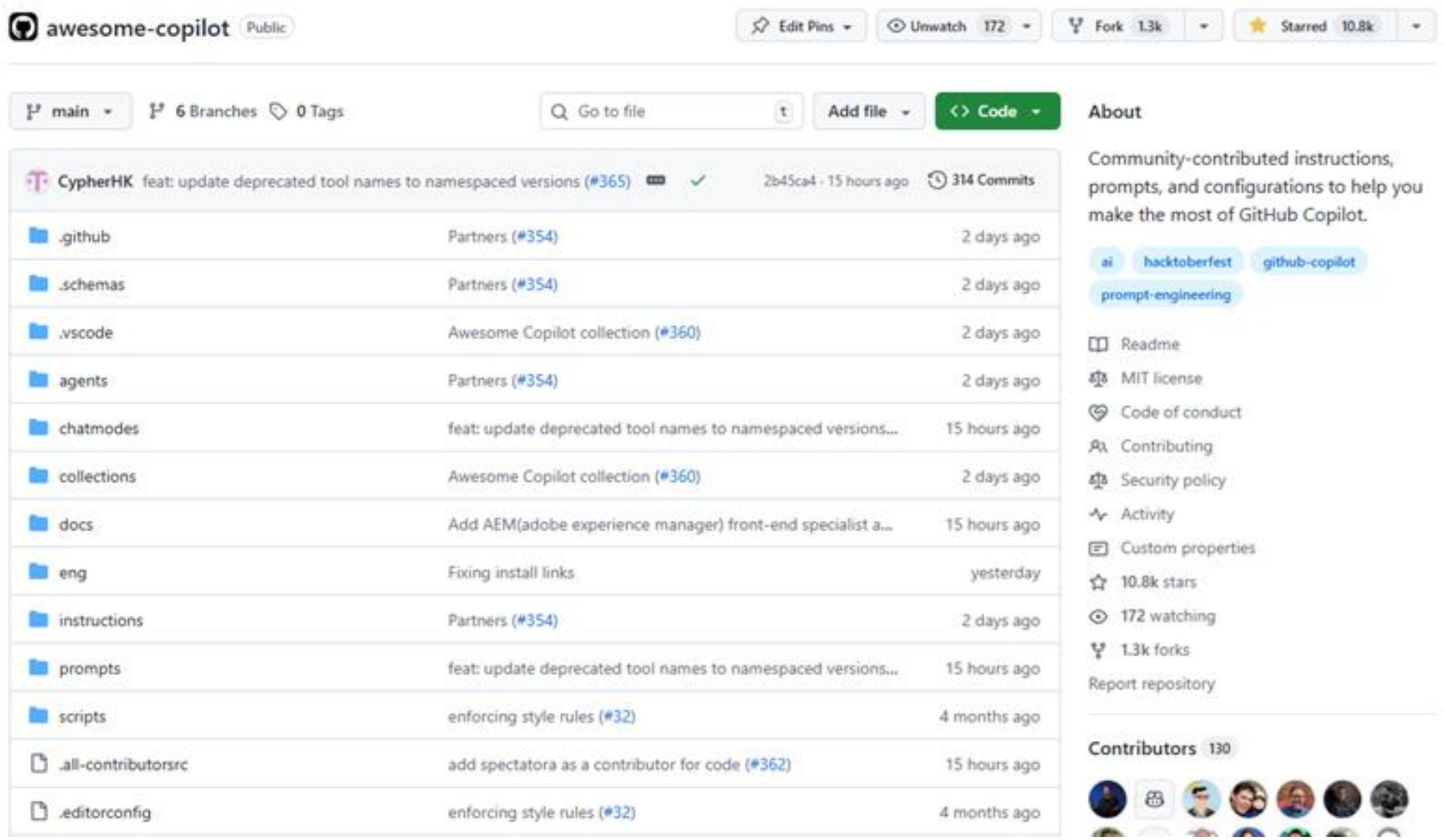
lab-soft
software engineering laboratory

Goals

- We aims to explore the extent to which variation in **prompt techniques** and their combinations affects **different refactoring types**.
- We intend to provide developers and tools with **practical insights** for creating refactoring prompts for real-world projects.



Relevance



awesome-copilot Public

Edit Pins · Unwatch 172 · Fork 1.3k · Starred 10.8k

main · 6 Branches · 0 Tags · Go to file · Add file · Code

CypherHK feat: update deprecated tool names to namespaced versions (#365) 2b45ca4 · 15 hours ago 314 Commits

.github	Partners (#354)	2 days ago
.schemas	Partners (#354)	2 days ago
.vscode	Awesome Copilot collection (#360)	2 days ago
agents	Partners (#354)	2 days ago
chatmodes	feat: update deprecated tool names to namespaced versions...	15 hours ago
collections	Awesome Copilot collection (#360)	2 days ago
docs	Add AEM(adobe experience manager) front-end specialist a...	15 hours ago
eng	Fixing install links	yesterday
instructions	Partners (#354)	2 days ago
prompts	feat: update deprecated tool names to namespaced versions...	15 hours ago
scripts	enforcing style rules (#32)	4 months ago
.all-contributorsrc	add spectatora as a contributor for code (#362)	15 hours ago
.editorconfig	enforcing style rules (#32)	4 months ago

About

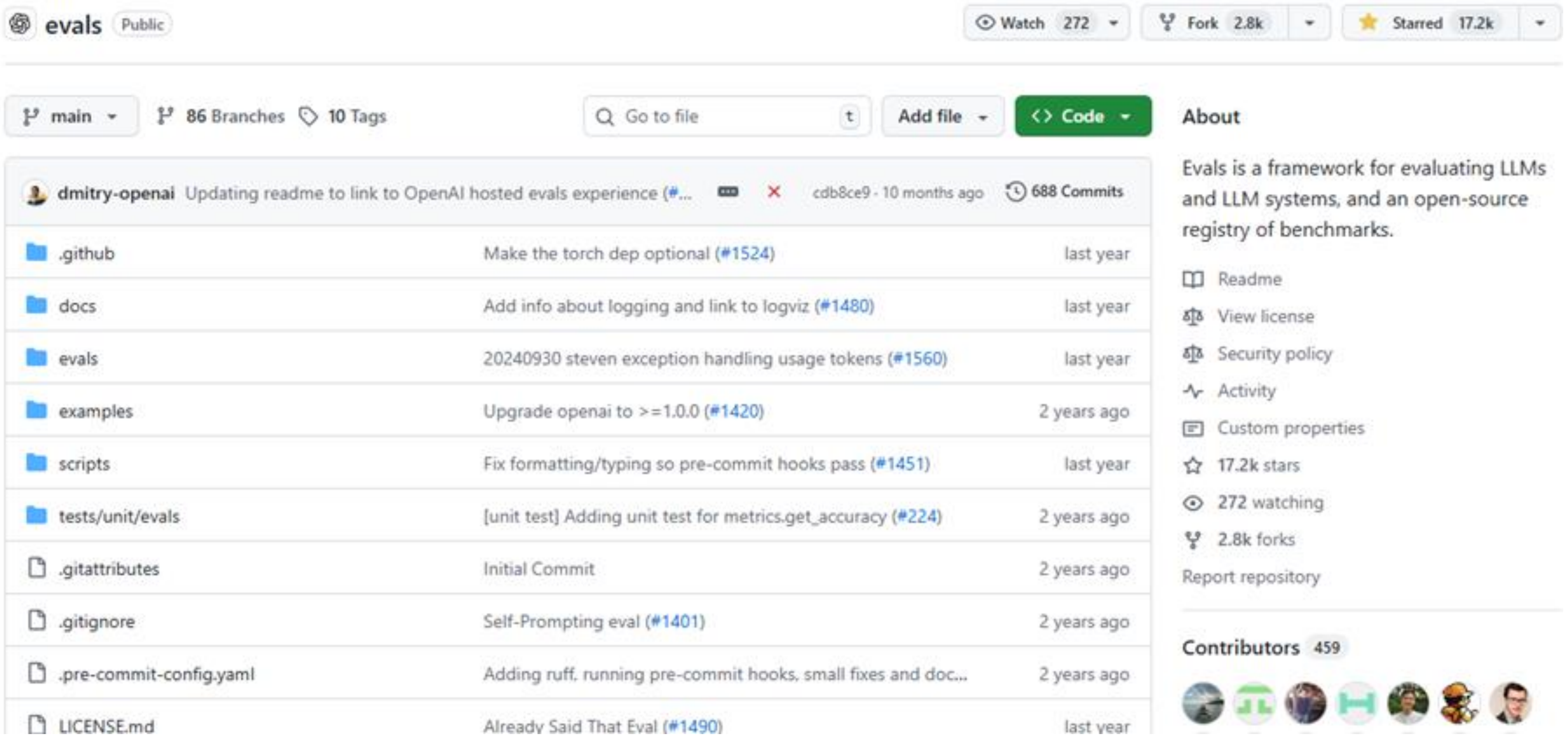
Community-contributed instructions, prompts, and configurations to help you make the most of GitHub Copilot.

ai · hacktoberfest · github-copilot · prompt-engineering

Readme
MIT license
Code of conduct
Contributing
Security policy
Activity
Custom properties
10.8k stars
172 watching
1.3k forks
Report repository

Contributors 130

Relevance



evals Public

Watch 272 | Fork 2.8k | Starred 17.2k

main | 86 Branches | 10 Tags | Go to file | Add file | Code

dmitry-openai Updating readme to link to OpenAI hosted evals experience (#... cdb8ce9 · 10 months ago 688 Commits

.github	Make the torch dep optional (#1524)	last year
docs	Add info about logging and link to logviz (#1480)	last year
evals	20240930 steven exception handling usage tokens (#1560)	last year
examples	Upgrade openai to >=1.0.0 (#1420)	2 years ago
scripts	Fix formatting/typing so pre-commit hooks pass (#1451)	last year
tests/unit/evals	[unit test] Adding unit test for metrics.get_accuracy (#224)	2 years ago
.gitattributes	Initial Commit	2 years ago
.gitignore	Self-Prompting eval (#1401)	2 years ago
.pre-commit-config.yaml	Adding ruff, running pre-commit hooks, small fixes and doc...	2 years ago
LICENSE.md	Already Said That Eval (#1490)	last year

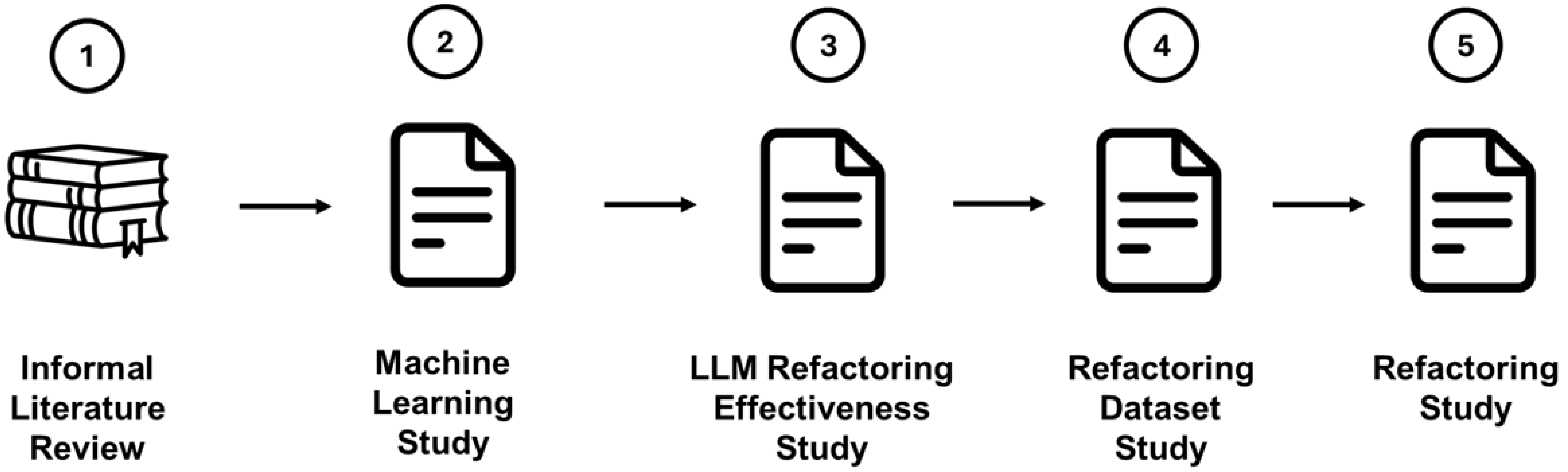
About

Evals is a framework for evaluating LLMs and LLM systems, and an open-source registry of benchmarks.

Readme
View license
Security policy
Activity
Custom properties
17.2k stars
272 watching
2.8k forks
Report repository

Contributors 459

Research Method


1
Informal Literature Review
2
Machine Learning Study
3
LLM Refactoring Effectiveness Study
4
Refactoring Dataset Study
5
Refactoring Study

Contribution Progress

- Nunes, H. G., Santana, A., Figueiredo, E., & Costa, H. *Tuning code smell prediction models: A replication study*. In International Conference on Program Comprehension (**ICPC 2024**) - **Study 1**

- Nunes, H. G., Figueiredo, E., Rocha, L., Nadi S., Ferreira F., & Esteves, G. *Evaluating the effectiveness of llms in fixing maintainability issues in real-world projects*. In International Conference on Software Analysis, Evolution and Reengineering (**SANER 2025**) - **Study 2**

- Nunes, H. G., Sharma, T., Figueiredo, E. *MaRV: A Manually Validated Refactoring Dataset*. In International Conference on AI Foundation Models and Software Engineering (**FORGE 2025**) - **Study 3**

Machine Learning to Detect Code Smells

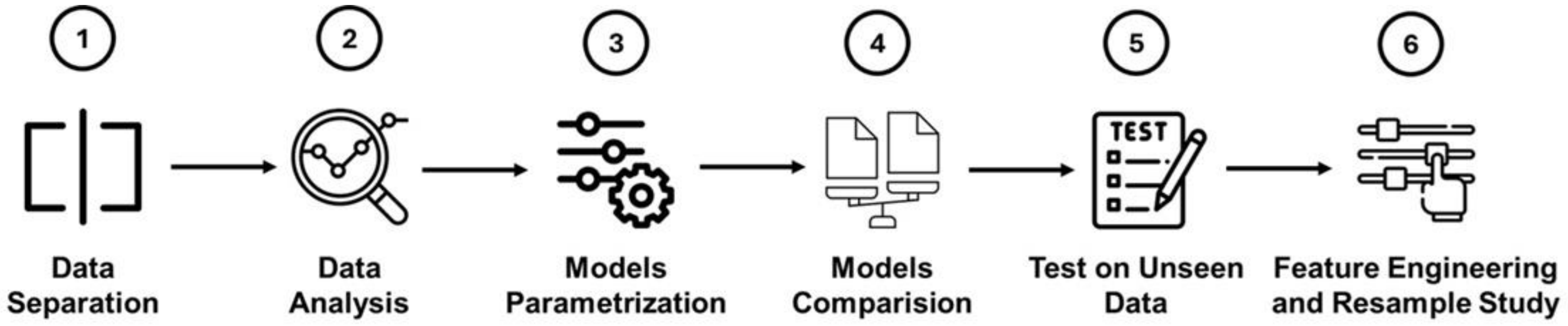
Study 1

lab·soft
software engineering laboratory

Study Context & Goals

- **Replicate** the study by Cruz et al. (2023), which uses machine learning to detect code smells, but **with a dataset of modern systems**.
- We use **seven traditional machine learning algorithms** to detect **four code smells**.
- We also evaluate how data resampling, feature selection, and polynomial feature techniques affect code smell predictions.

Design of Study 1



lab-soft
software engineering laboratory

Study Result & Conclusion

- The performance of traditional ML algorithms for code smell detection are **limited**.
- ML techniques, especially resampling, improve prediction performance, but **not enough**.
- **This study indicated that we should explore for other solutions. LLMs?**

lab-soft
software engineering laboratory

Large Language Models for Code Refactoring

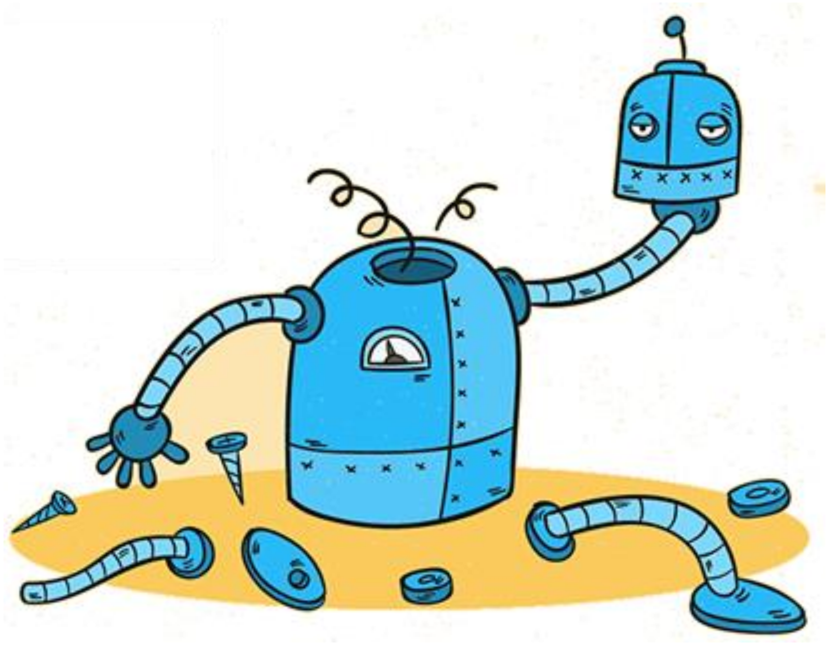
Study 2

lab·soft

software engineering laboratory

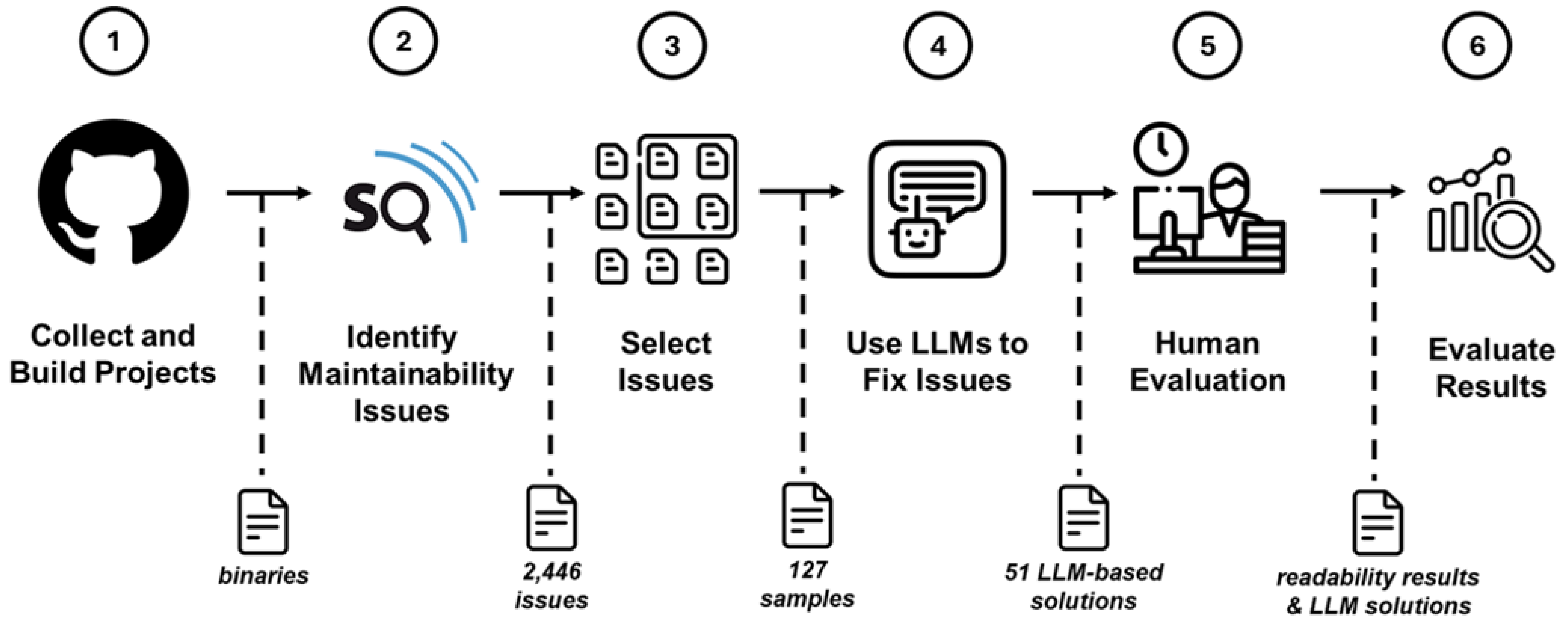
Study Context & Goals

- We evaluated the **effectiveness of LLMs for refactoring** 10 issue categories in real-world projects.
- Furthermore, the study assessed the most common **LLM failures**.
- We also conducted a human **readability evaluation**.

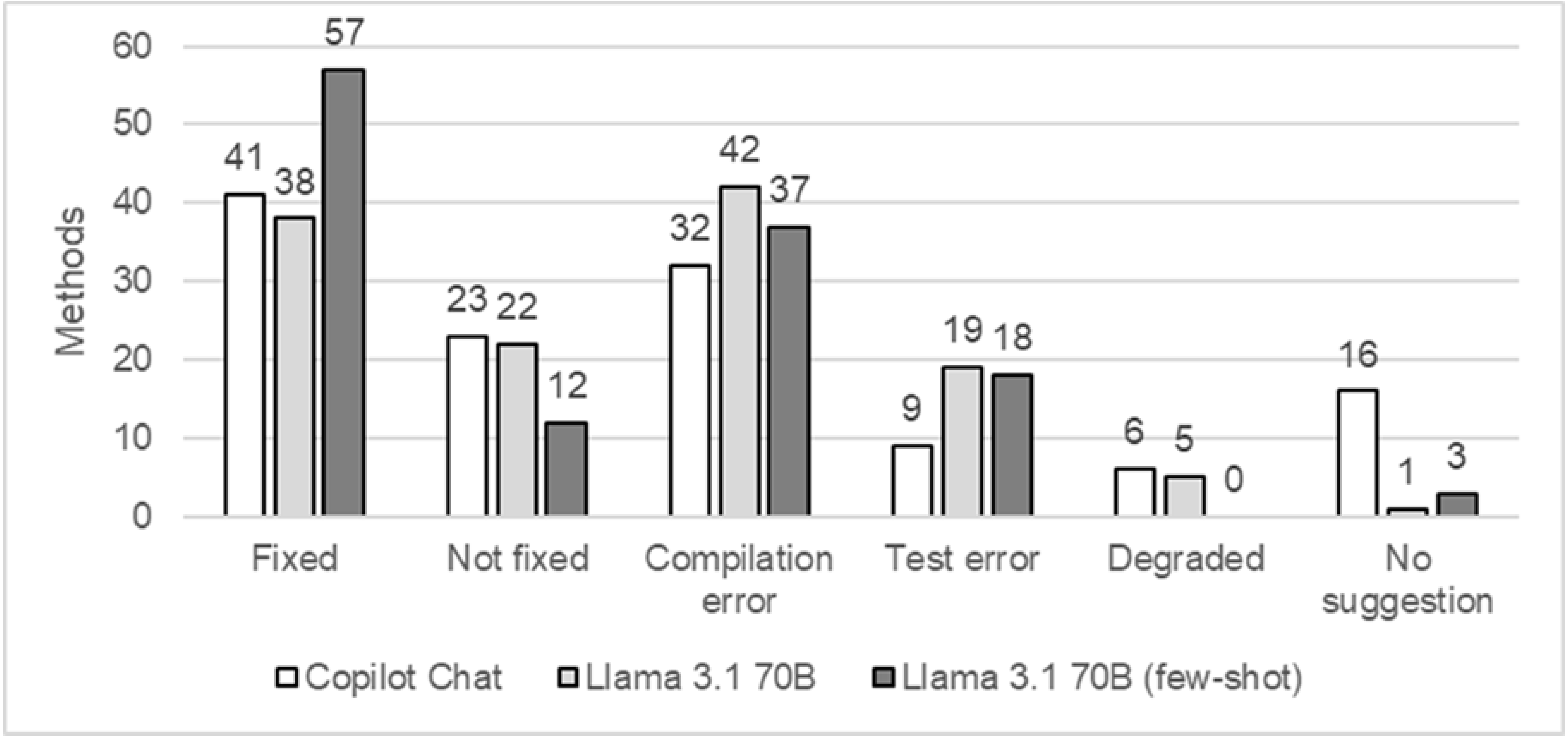


lab·soft
software engineering laboratory

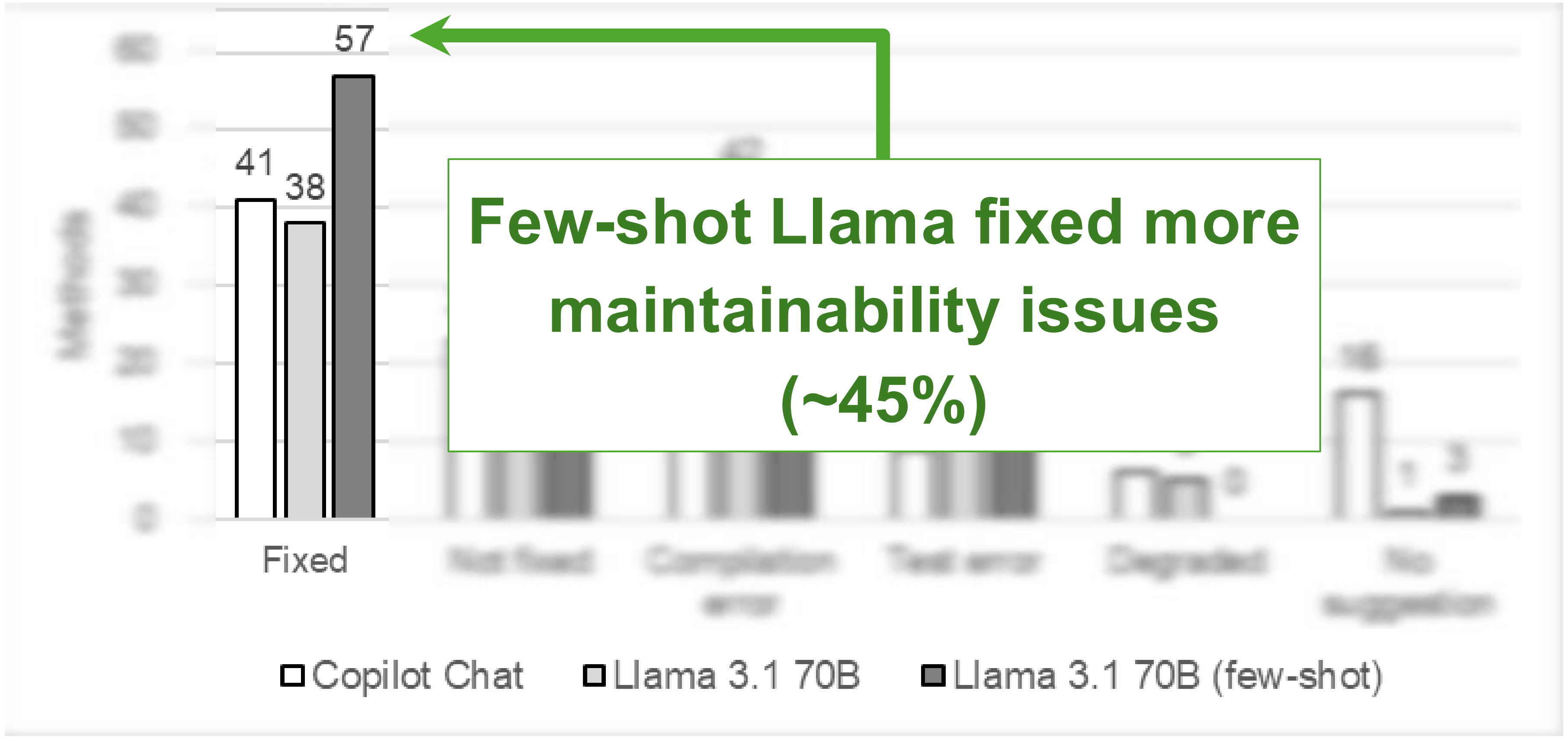
Design of Study 2



Effectiveness of the LLM

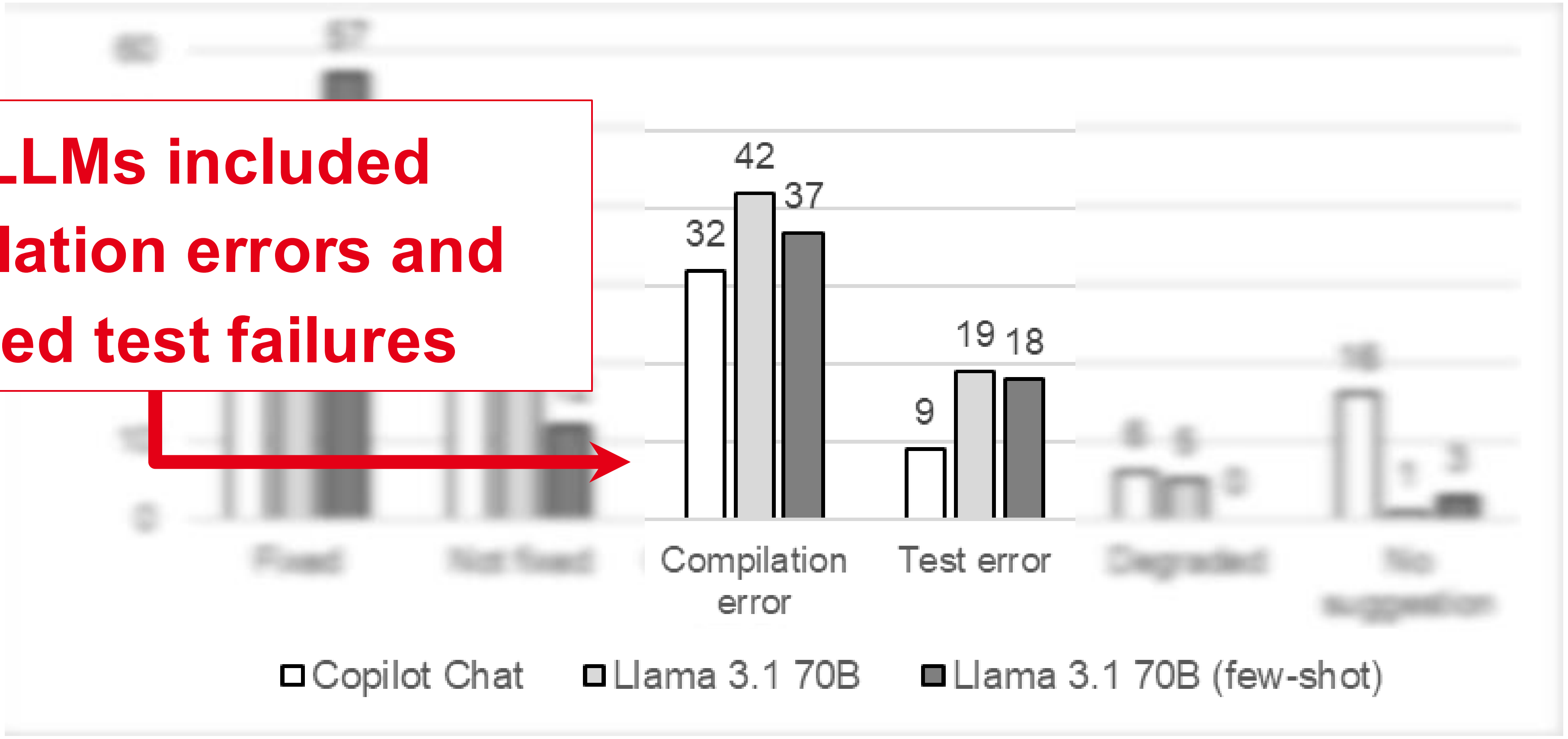


Effectiveness of the LLM



Few-shot Llama fixed more maintainability issues (~45%)

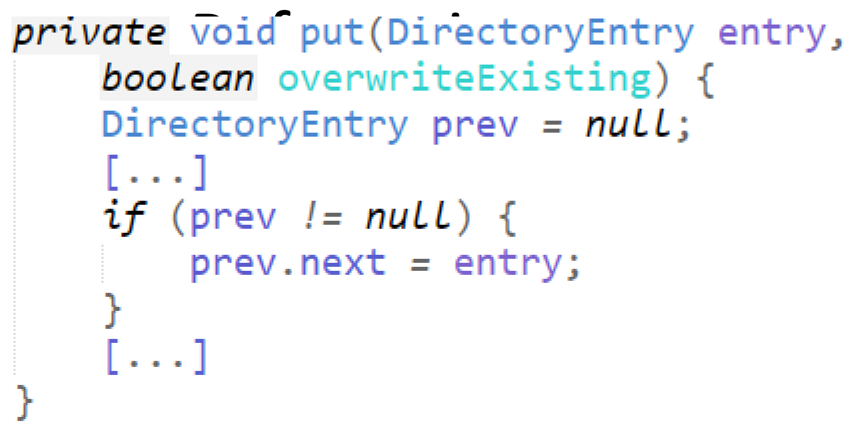
Effectiveness of the LLM



All LLMs included compilation errors and caused test failures

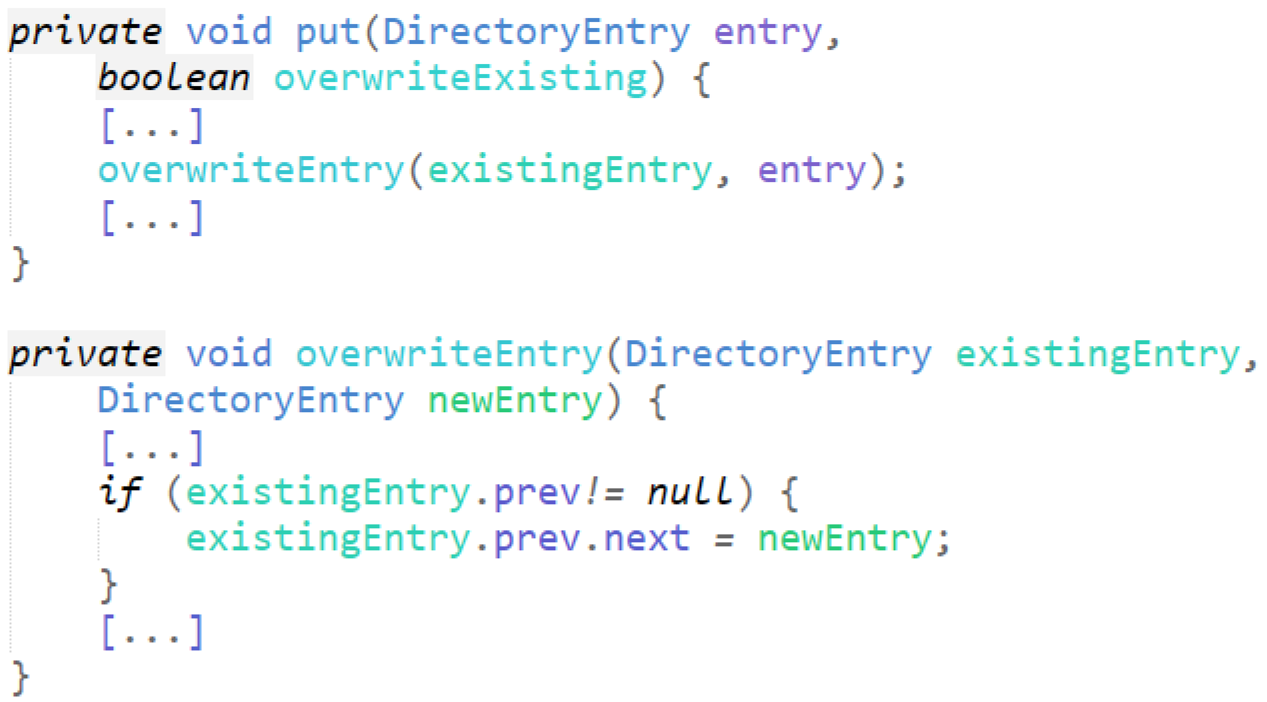
LLM Failures (Hallucination)

Original



```
private void put(DirectoryEntry entry,
    boolean overwriteExisting) {
    DirectoryEntry prev = null;
    [...]
    if (prev != null) {
        prev.next = entry;
    }
    [...]
}
```

LLM



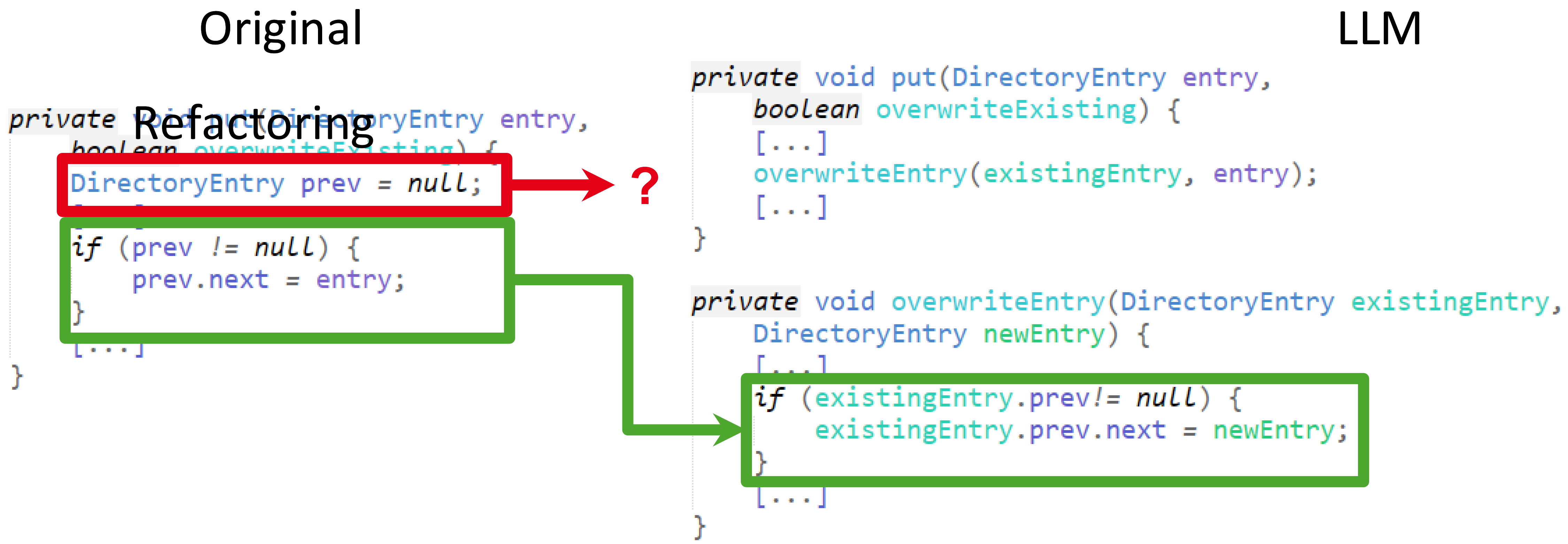
```
private void put(DirectoryEntry entry,
    boolean overwriteExisting) {
    [...]
    overwriteEntry(existingEntry, entry);
    [...]
}

private void overwriteEntry(DirectoryEntry existingEntry,
    DirectoryEntry newEntry) {
    [...]
    if (existingEntry.prev!= null) {
        existingEntry.prev.next = newEntry;
    }
    [...]
}
```

lab-soft

software engineering laboratory

LLM Failures (Hallucination)



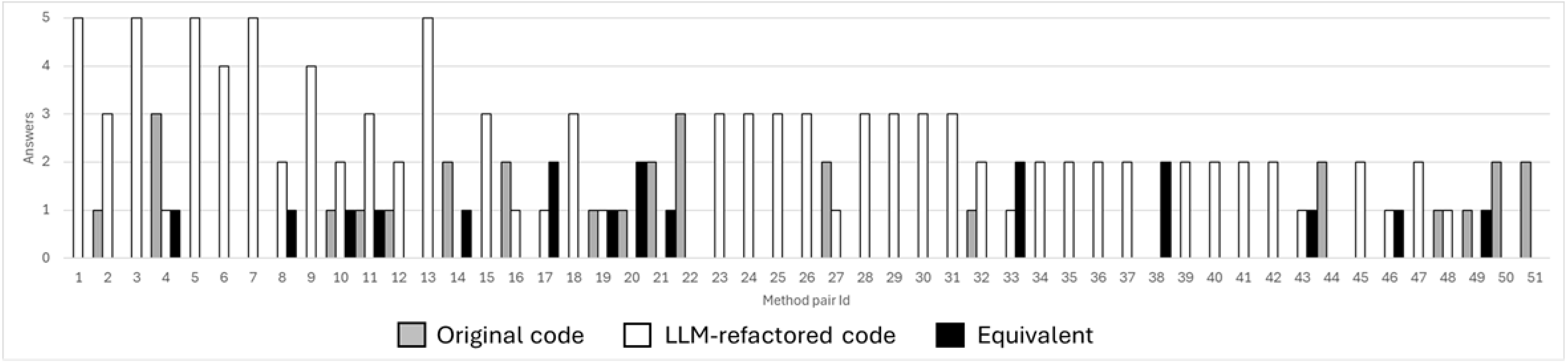
```
private void put(DirectoryEntry entry,
    boolean overwriteExisting) {
    DirectoryEntry prev = null;
    [...]
    if (prev != null) {
        prev.next = entry;
    }
    [...]
}
```

```
private void put(DirectoryEntry entry,
    boolean overwriteExisting) {
    [...]
    overwriteEntry(existingEntry, entry);
    [...]
}

private void overwriteEntry(DirectoryEntry existingEntry,
    DirectoryEntry newEntry) {
    [...]
    if (existingEntry.prev!= null) {
        existingEntry.prev.next = newEntry;
    }
    [...]
}
```

Human Evaluation of LLM Readability (RQ3)

Out of 51 method pairs, **~70%** considered LLM-refactored code more readable.



Study Conclusion

- Despite **limited** effectiveness (below 50%), **few shot learning** presented the highest effectiveness (~40%), if compared with zero shot learning (~30%).
- Overall, participants considered LLM-refactored code **more readable**.
- **The results of this study motivated us to evaluate more deeply the use of prompt techniques for code refactoring.**

Refactoring Dataset

Study 3

lab·soft

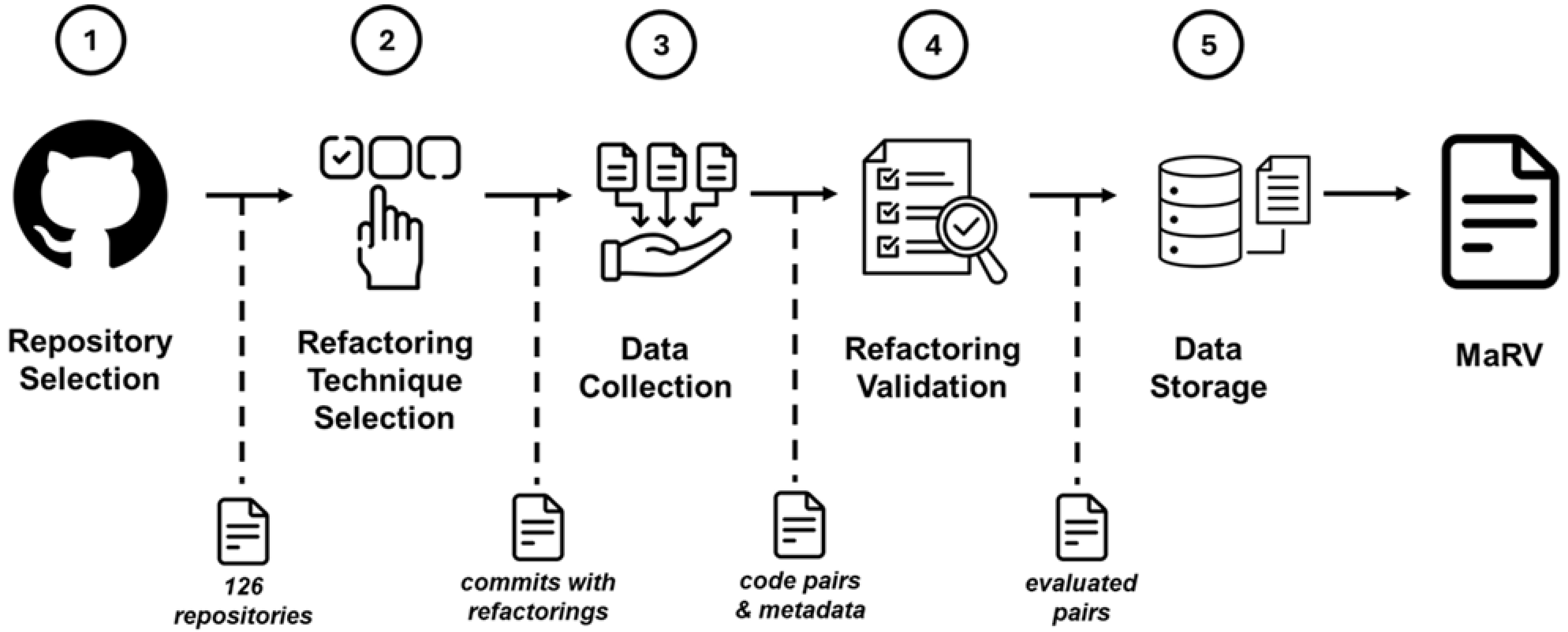
software engineering laboratory

Study Context & Goal

- We used RefactoringMiner to **collect refactorings** and conducted a human evaluation to **select the most representative examples**.
- We aim to produce a high-quality **manually validated dataset** of actual refactorings from open-source projects, **MaRV**.

lab·soft
software engineering laboratory

Design of Study 3



Manual Evaluation Results

	Votes	Count	Total
Consensus	[disagree, disagree]	84	405
	[agree, agree]	321	
Conflict	[disagree, agree]	217	285
	[I don’t know, disagree]	27	
	[I don’t know, agree]	41	
Other	[I don’t know, I don’t know]	3	3
			693

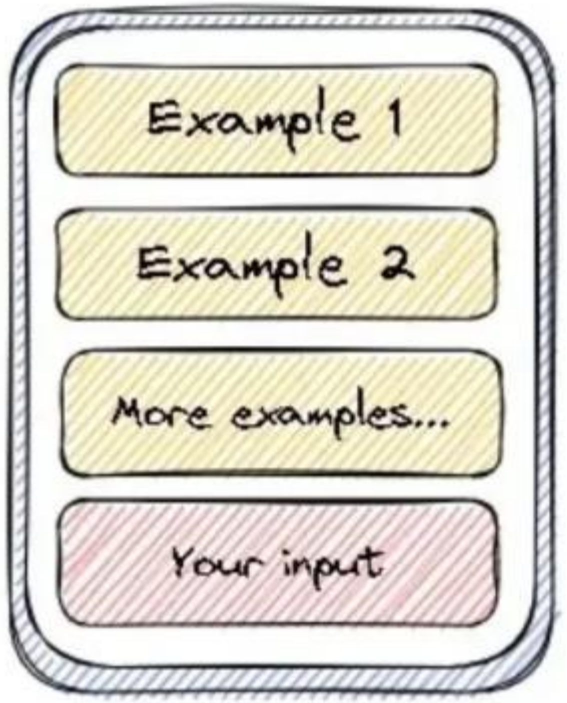
Manual Evaluation Results

	Votes	Count	Total
Consensus	[disagree, disagree]	84	405
	[agree, agree]	**321**	
Conflict	[disagree, agree]	217	285
	[I don't know, disagree]	27	
	[I don't know, agree]	41	
Other	[I don't know, I don't know]	3	3
			693

Both participants agreed in 321 cases (~46%). We considered this cases for MaRV

Study Conclusion

- **We aim to use MaRV to support several prompting techniques (e.g., few-shot prompting).**



lab-soft
software engineering laboratory

Agenda

Next Steps

lab·soft

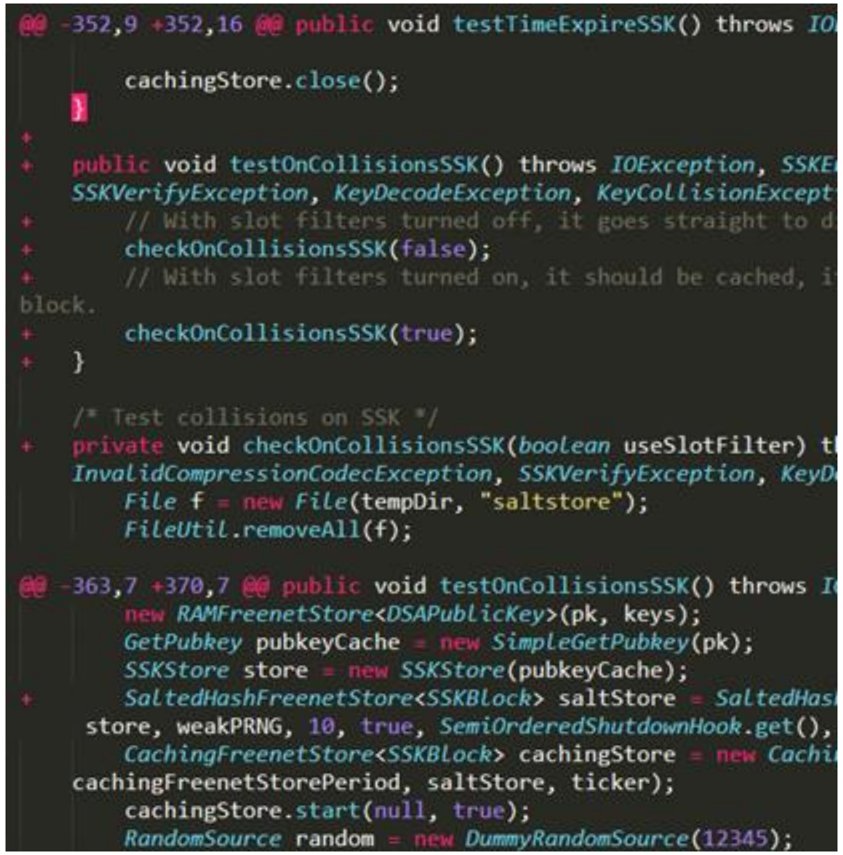
software engineering laboratory

Future Steps

1. Improve MaRV Dataset
2. Select LLMs and Prompting Techniques
3. Define the Prompt Design
4. Execute and Validate LLM-Based Refactorings
5. Evaluate Results and Consolidate in a Paper
6. Extend our LLM Study
7. Write and Submit a Journal Paper
8. Write and Present Thesis

lab·soft
software engineering laboratory

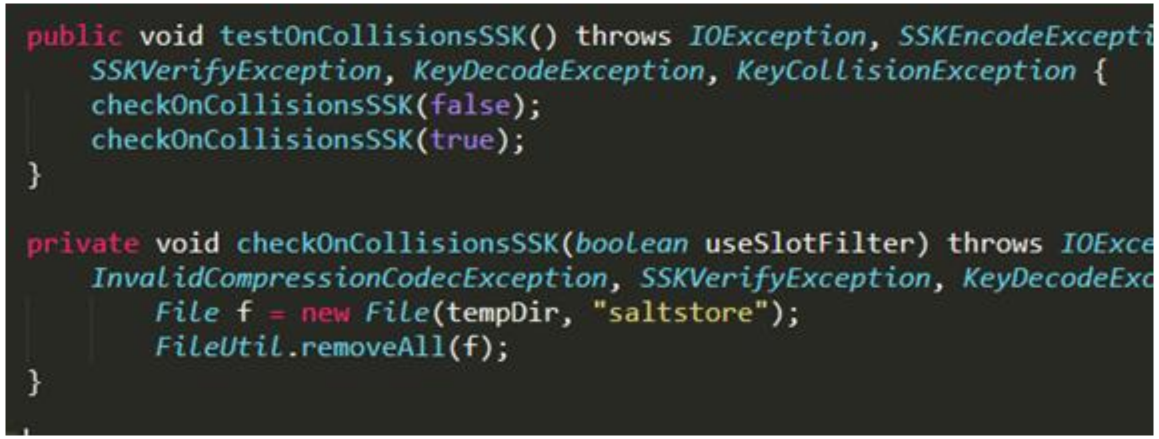
Improve MaRV



```
@@ -352,9 +352,16 @@ public void testTimeExpireSSK() throws IO
          cachingStore.close();
      }
+
+     public void testOnCollisionsSSK() throws IOException, SSKE
      SSKVerifyException, KeyDecodeException, KeyCollisionExcept
+         // With slot filters turned off, it goes straight to d
+         checkOnCollisionsSSK(false);
+         // With slot filters turned on, it should be cached, i
block.
+         checkOnCollisionsSSK(true);
+     }

      /* Test collisions on SSK */
+     private void checkOnCollisionsSSK(boolean useSlotFilter) t
      InvalidCompressionCodecException, SSKVerifyException, KeyD
          File f = new File(tempDir, "saltstore");
          FileUtil.removeAll(f);

@@ -363,7 +370,7 @@ public void testOnCollisionsSSK() throws I
          new RAMFreenetStore<DSAPublicKey>(pk, keys);
          GetPubkey pubkeyCache = new SimpleGetPubkey(pk);
          SSKStore store = new SSKStore(pubkeyCache);
+         SaltedHashFreenetStore<SSKBlock> saltStore = SaltedHas
       store, weakPRNG, 10, true, SemiOrderedShutdownHook.get(),
          CachingFreenetStore<SSKBlock> cachingStore = new Cachi
      cachingFreenetStorePeriod, saltStore, ticker);
          cachingStore.start(null, true);
          RandomSource random = new DummyRandomSource(12345);
```

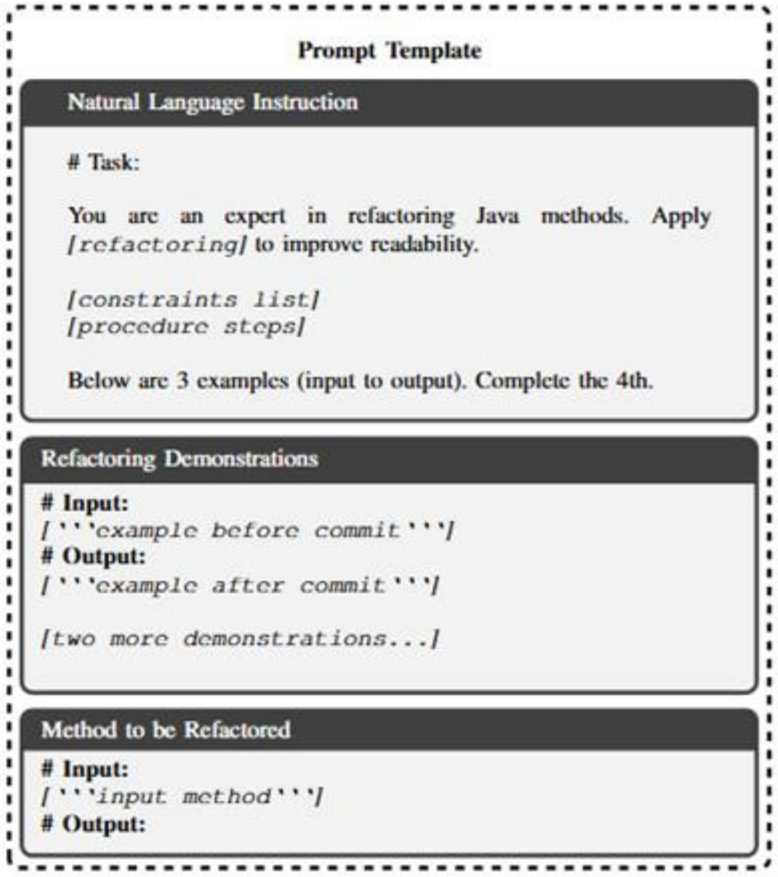


```
public void testOnCollisionsSSK() throws IOException, SSKEncodeExcepti
    SSKVerifyException, KeyDecodeException, KeyCollisionException {
    checkOnCollisionsSSK(false);
    checkOnCollisionsSSK(true);
}

private void checkOnCollisionsSSK(boolean useSlotFilter) throws IOExce
    InvalidCompressionCodecException, SSKVerifyException, KeyDecodeExc
        File f = new File(tempDir, "saltstore");
        FileUtil.removeAll(f);
}
```

Prompt Evaluation

- We aim to evaluate **different prompt techniques** and their variations.
- We will define several prompt designs, **varying and combining techniques**.
- We want to identify **which prompts are most effectiveness** for each different refactoring type.



Prompt Template

Natural Language Instruction

Task:

You are an expert in refactoring Java methods. Apply *[refactoring]* to improve readability.

[constraints list]
[procedure steps]

Below are 3 examples (input to output). Complete the 4th.

Refactoring Demonstrations

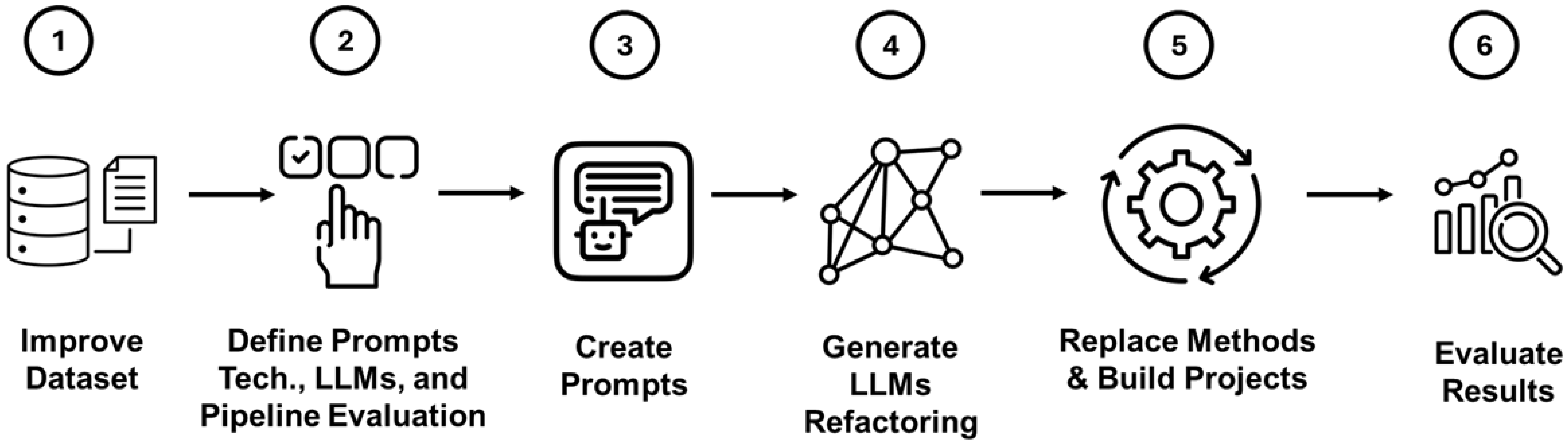
# Input:
['''example before commit''']
# Output:
['''example after commit''']

[two more demonstrations...]

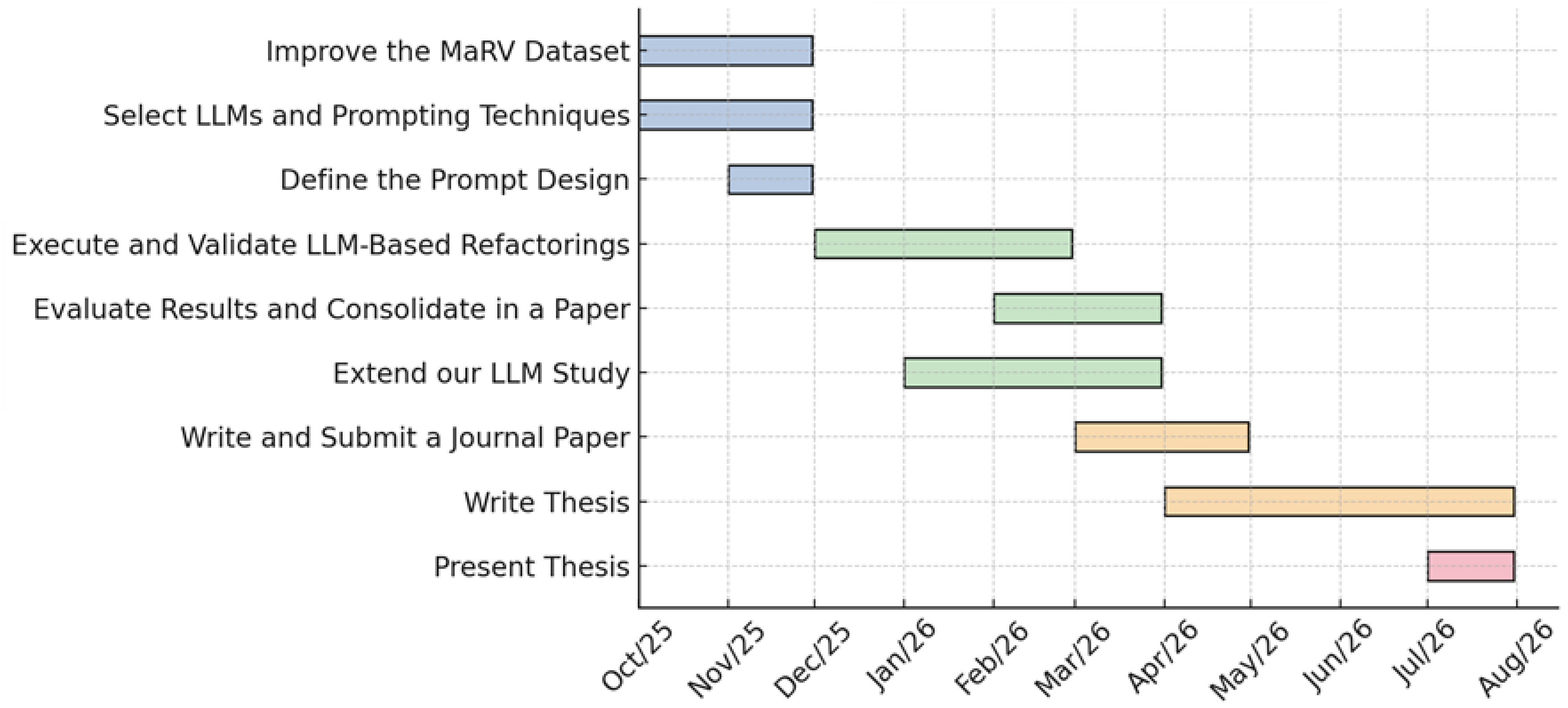
Method to be Refactored

# Input:
['''input method''']
# Output:

Next Steps Overview


1
2
3
4
5
6
Improve Dataset
Define Prompts Tech., LLMs, and Pipeline Evaluation
Create Prompts
Generate LLMs Refactoring
Replace Methods & Build Projects
Evaluate Results

Agenda



Improve the MaRV Dataset
Select LLMs and Prompting Techniques
Define the Prompt Design
Execute and Validate LLM-Based Refactorings
Evaluate Results and Consolidate in a Paper
Extend our LLM Study
Write and Submit a Journal Paper
Write Thesis
Present Thesis
Oct/25
Nov/25
Dec/25
Jan/26
Feb/26
Mar/26
Apr/26
May/26
Jun/26
Jul/26
Aug/26

Thank you!

Henrique Nunes | henrique.mg.bh@gmail.com

lab·soft
software engineering laboratory