Readability - Driven Test Selection

Poor software quality in the U.S. cost **$2.41 trillion** in 2022 [1], underscoring the need for better practices. Although writing tests improves the code quality [2], developers often neglect writing tests. Existing tools generate tests automatically but **struggle with readability.**

This research aims to **enhance readability** by creating a quantifiable metric, using Natural Language Processing (**NLP**), readability factors and Large Language Models (**LLMs**) to automate the evaluation process.

01 Research Question

Research Question: How can LLMs be utilized to assign readability scores and rank automatically generated unit tests based on their readability?

Subquestions:
SQ1: How can existing readability metrics be adapted to develop an LLM-based algorithm for evaluating unit test code?

SQ2: How accurately can LLMs assess the readability of unit tests compared to human evaluations?

How does this research **contribute to existing research?** This research contributes to the existing literature by introducing an approach to improving the readability of automatically generated tests **by developing and validating a readability metric,** it bridges the gap between high test coverage and low comprehensibility.

02 Background

Automatic test generation tools have become increasingly sophisticated. However, a significant issue with these tools is the **readability of the generated tests.** The primary reason is that the generated identifiers, test names, and associated data are **not based on the context**, making them difficult to interpret [6]. Existing readability metrics focus primarily on the **structural aspects** of the code, such as line length, number of assertions, and identifier length, without considering the contextual relevance [3], [4], [5]. As a result, tests like the one shown in Figure 1 may receive high readability scores despite being hard to understand. **Context is crucial** for true readability. Large Language Models (LLMs) have the capability to **analyze and understand the context** of code. Leveraging LLMs, we can enhance the evaluation of code readability by incorporating contextual understanding.

03 Implementation

To evaluate code readability, we focused on two different aspects: **specific aspects** and **general aspects.** For these evaluations, we developed prompts that allowed a LLM to score the code.

Feature Selection
- **Specific Aspects**: These include ranking of identifiers, test names, and comments.
- **General Aspects**: These include ranking of conciseness, completeness, and naturalness.

We created three types of prompts to guide the LLM in scoring:
- **Baseline Prompt:** This prompt asks the LLM to rank the code without any guidelines, see Figure 2.
- **Simple Prompt:** This prompt provides a scoring scale from 1 to 5 and explains the criteria for each score, see Figure 3.
- **Few-Shot Prompt:** This prompt not only provides the scoring scale but also includes examples of good and bad code, explaining how to rate them.

```
Please follow these steps:
1. Carefully read the Java code provided between the [CODE] tags.
2. Assess the readability of the code on a scale from 1 to 5, where 1 means
↪  very difficult to understand and 5 means very easy to understand.
3. Indicate your readability score by placing it between the [SCORE] and
↪  [/SCORE] tags.
Your goal is to provide an evaluation solely focused on the readability of
↪  the code.[/INST]
```

Fig 2, Baseline Prompt

```
@Test
@Timeout(value = 4000, unit = TimeUnit.MILLISECONDS)
public void tstXY12A45() throws Throwable {
    JSWeaponData abc123 = new JSWeaponData();
    SimpleWeapon def456 = new SimpleWeapon(-2704, "3]B;R`L#}d:!",
    boolean ghi789 = abc123.isGoldenGun(def456);
    assertFalse(ghi789);
}
```

Fig 1, Motivating Example

```
b. Test Name: Assess the descriptiveness and clarity of test method name.
  1: Test name is vague or does not reflect the purpose of the test (e.g.,
  ↪  test1).
  2: Test name is somewhat unclear or generic.
  3: Test name is fairly descriptive but could be more specific.
  4: Test name is clear and mostly descriptive.
  5: Test name is highly descriptive and clearly reflects the purpose of
  ↪  the tests.
```

Fig 3, Simple PromptEvoSuite

04 User Evaluation

Objective: Compare human and LLM code readability scores to evaluate alignment and accuracy, specifically focusing on how humans rate the readability of specific and general aspects of code snippets.

User Evaluations:
- Participants: 11 CS students (8 BSc, 3 MSc) from TU Delft.
- Experience: 4 with 0-1 years, 1 with 1-3 years, 6 with 3-5 years of Java testing.
- Procedure:
  - Evaluate 10 code snippets (3 from EvoSuite, 3 from UTGen, 4 from public repositories).
  - Provide overall readability scores (1-5) and specific and general aspect ratings.
  - Rank importance of aspects.

Result:
Table of readability scores for specific and general aspects of the code, see Figure 6.

06 Conclusion

This study demonstrated that LLMs, especially when using simple prompts with OpenAI's GPT models, can **effectively evaluate and rank the readability** of automatically generated unit tests. The strong alignment between LLM scores and human judgments highlights the potential of LLMs in this domain, showing promise for future research to further refine and expand their applications.

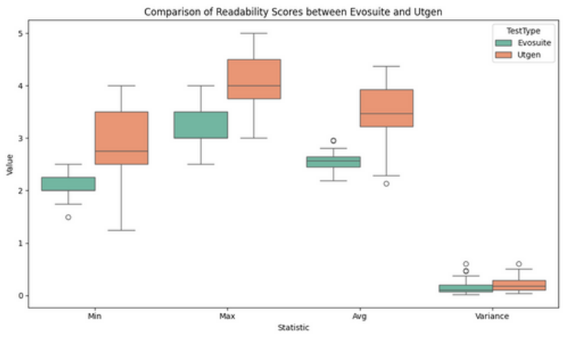

Fig 4, EvoSuite vs UTGen

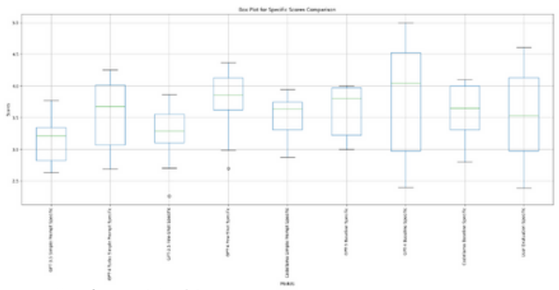

Fig 5, Specific Aspect Boxplot

05 Results

SQ1: LLM-based Algorithm for Scoring Readability
- See Figure 4

SQ2: LLM Evaluations vs. Human Evaluations
- Results Summary:
  - **Specific Aspects:** GPT-4 Turbo Simple Prompt model performed best with an MAE (Mean Absolute Error) of 0.3754, RMSE (Root Mean Squared Error) of 0.4607, and correlation of 0.7632 (see Figure 5)
  - **General Aspects:** GPT-3.5 Simple Prompt model performed best with an MAE of 0.2886, RMSE of 0.3583, and correlation of 0.8021.
  - **Baseline Models:** Codellama Baseline had the best performance with an MAE of 0.4420, RMSE of 0.5705, and correlation of 0.6089.

The high correlation between LLM and human evaluations underscores the reliability of our LLM-based readability scoring, demonstrating that **LLMs can effectively replicate human judgments** in assessing test case readability.

07 Future Work

Future Work:
1. **Expand to Other Test Types:** Apply the readability evaluation approach to other types of tests and programming languages.
2. **Broader Dataset and LLM Configurations**: Use a wider range of datasets and LLM configurations to enhance generalizability.
3. **Integration into Test Generation Systems**: Explore integrating LLM-based readability rankings into automated test generation systems.
4. **Adjusting Metric Weights**: Investigate using different weights for readability metrics to tailor assessments better.

TestType	TestName	ScoreType	Avg	Overall
EvoSuite	Easy	specific	2.97	4.02
EvoSuite	Easy	general	3.97	4.02
EvoSuite	Medium	specific	2.72	2.85
EvoSuite	Medium	general	3.33	2.85
EvoSuite	Hard	specific	3.00	3.45
EvoSuite	Hard	general	3.52	3.45
UTGEN	Easy	specific	4.61	4.83
UTGEN	Easy	general	4.73	4.83
UTGEN	Medium	specific	4.37	4.42
UTGEN	Medium	general	4.24	4.42
UTGEN	Hard	specific	4.27	4.45
UTGEN	Hard	general	4.36	4.45
OtherTests	Test1	specific	2.39	2.08
OtherTests	Test1	general	2.85	2.08
OtherTests	Test2	specific	3.70	3.95
OtherTests	Test2	general	4.06	3.95
OtherTests	Test3	specific	3.73	3.98
OtherTests	Test3	general	3.82	3.98
OtherTests	Test4	specific	3.37	3.57
OtherTests	Test4	general	3.32	3.57

Fig 6, User Evaluation Results

Authors
Ismaël Zaidi

Supervisor
Amir Deljouyi,
Andy Zaidman

Affiliations
Technical University Delft, CSE3000

References

[1] Consortium for Information & Software Quality (CISQ). The Cost of Poor Quality Software in the US: A 2022 Report. Accessed: 2023-04-22. 2022. url: https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2022-report
[2] L. Crispin. "Driving Software Quality: How Test-Driven Development Impacts Software Quality". In: IEEE Software 23 (2006). doi: 10.1109/MS.2006.157
[3]E. Daka, J. Campos, G. Fraser, J. Dorn, and W. Weimer, "Modeling readability to improve unit tests," Aug. 2015, doi: https://doi.org 10.1145/2786805.2786838.
[4] E. Daka "Improving Readability of Automatically Generated Unit Tests", 2015
[5] D. Winkler, P. Urbanke and R. Ramler, "What Do We Know About Readability of Test Code? - A Systematic Mapping Study," 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Honolulu, HI, USA, 2022, pp. 1167-1174, doi: 10.1109/SANER53432.2022.00135.
[6] Giovanni Grano, Simone Scalabrino, H. Gall, and R. Oliveto. An empirical investigation on the readability of manual and generated test cases. 2018 IEEE/ACM
26th International Conference on Program Comprehension (ICPC), pages 348–3483, 2018.
[7] Paper has not yet been published