NoseSense: Benchmarking LLMs for Test Smell Detection

Magno Macedo
Instituto de Computação
Universidade Federal da Bahia (UFBA)
Salvador, Brasil
magnomiranda@ufba.br

Tássio Virgínio
Instituto de Computação
Universidade Federal da Bahia (UFBA)
Salvador, Brasil
tassio.virginio@ufba.br

Ivan Machado
Instituto de Computação
Universidade Federal da Bahia (UFBA)
Salvador, Brasil
ivan.machado@ufba.br

Abstract

Unit test quality is a critical concern in software engineering, as anti-patterns in test code (known as test smells) compromise maintainability and reliability of test suites. The rapid evolution of Large Language Models (LLMs) has motivated their application to this domain, yet the pace of model releases renders point-in-time empirical studies quickly outdated. In this paper, we develop NoseSense, a benchmark designed to enable systematic and reproducible evaluation of LLMs for test smell detection. We mined 1,275 open-source Java repositories, applied the JNose static analysis tool to curate a balanced dataset of 1,100 real test smell instances, and evaluated six state-of-the-art models through a dedicated automated tool. The results indicate that model scale and generation significantly impact performance: Gemma-4-31B-it achieved the highest accuracy (67.36%), whereas the smaller-scale Gemma-3n-E4B-it achieved 38.55%. NoseSense provides a robust and reproducible foundation for monitoring LLM capabilities in software quality assurance, with a publicly available dataset, replication package, and benchmark implementation.

Keywords

LLMs, Benchmarking, Test Smells, Software Testing.

Resumo. A rápida evolução dos Large Language Models (LLMs) torna avaliações empíricas estáticas rapidamente obsoletas. Este artigo apresenta o NoseSense, um benchmark automatizado e reprodutível para avaliar a capacidade de LLMs na detecção de test smells em testes de unidade Java. O benchmark combina um Gold Set curado com 1.100 métodos de teste reais, abrangendo dez categorias de test smells e testes limpos, com um framework de avaliação automatizado. Para validar o benchmark, avaliamos seis LLMs estado-da-arte das famílias GPT-OSS, Gemma e Qwen. Os resultados revelaram diferenças substanciais de desempenho: o Gemma-4-31B-it obteve a maior acurácia global (67,36%) e o Gemma-3n-E4B-it a menor (38,55%). Melhorias geracionais nos modelos parecem ter maior impacto na detecção de test smells do que o aumento de escala isolado. Com o benchmark, dataset e pacote de replicação disponíveis publicamente, o NoseSense oferece uma base reutilizável para avaliar sistematicamente futuras gerações de LLMs em tarefas de teste de software.

1 Introduction

The rapid evolution of LLMs [2, 5] has motivated their application in Software Engineering tasks, including software quality and testing [6, 7]. However, static point-in-time evaluations quickly become outdated as newer model generations are released. Benchmarks that integrate tasks, metrics, and datasets into unified evaluation frameworks are essential to address this challenge [2].

Test smells are anti-patterns in unit test code that may compromise test suite maintainability and effectiveness [1]. While static analysis tools such as JNose [8] detect test smells via deterministic, rule-based AST analysis, no automated benchmark existed to systematically evaluate LLMs' semantic reasoning capabilities on this task.

This paper presents NoseSense, an automated and extensible benchmarking platform for evaluating LLMs on test smell detection. The main contributions are: (i) a curated Gold Set of 1,100 Java test methods labeled across ten test smell categories and clean tests; (ii) an automated evaluation pipeline that orchestrates multi-model experiments at scale; and (iii) a publicly available replication package enabling reproducible and longitudinal comparisons across successive LLM generations. The study addresses three research questions: RQ1. How effective are state-of-the-art LLMs in identifying and classifying test smells? RQ2. How do different LLMs compare in their overall performance and in specific types of test smells? RQ3. What is the relationship between model scale and performance in detecting test smells?

2 Methodology

The benchmark construction followed four phases. Phase 1 (Repository Collection): We selected nine GitHub organizations, identifying 1,473 repositories. These repositories were then filtered to determine whether they contained test files, yielding 1,275 Java projects. Phase 2 (Oracle Generation): JNose [8] was applied to all collected test files to generate the ground-truth labels for each test method. From this analysis, we retained ten test smell categories with sufficient occurrence frequency: Assertion Roulette, Magic Number Test, Eager Test, Ignored Test, Unknown Test, Verbose Test, Conditional Test Logic, Duplicate Assert, Exception Catching/Throwing, and Sensitive Equality. Phase 3 (Gold Set Curation): Using uniform stratified sampling, we selected 100 test methods per category plus 100 clean tests, totaling 1,100 benchmark instances. Only test methods containing a single smell type (or none) were retained to avoid ambiguous multi-label cases. Phase 4 (Automated Evaluation via NoseSense): The NoseSense tool orchestrates the experiment through a decoupled client-server architecture: a React frontend and a Python/FastAPI backend that manages asynchronous API calls to LLM providers via LangChain. Prompts were structured using the CRISPE framework [4], presenting each test snippet as a multiple-choice question with four alternatives (A–D dynamically randomized plus a fixed "None" option). The structured answer format `{Letter}` enabled automated response parsing at scale.

We evaluated six state-of-the-art LLMs from three families selected for provider diversity, scale diversity, and architectural diversity: Gemma-4-31B-it and Gemma-3n-E4B-it (Google Gemma, different generations), GPT-OSS-120B and GPT-OSS-20B (GPT-OSS, isolating the scale variable), and Qwen-3-235B (MoE) and Qwen-3.5-9B (dense), contrasting architectures within the Alibaba Qwen family.

3 Results

Table 1 summarizes the overall accuracy of each evaluated model. Gemma-4-31B-it achieved the highest accuracy (67.36%), while Gemma-3n-E4B-it obtained the lowest (38.55%).

Table 1: Overall Performance of the Evaluated LLMs

Model	Correct / Total	Accuracy (%)
Gemma-4-31B-it	741 / 1100	67.36%
GPT-OSS-120B	683 / 1100	62.09%
GPT-OSS-20B	646 / 1100	58.73%
Qwen-3-235B	572 / 1100	52.00%
Qwen-3.5-9B	531 / 1100	48.27%
Gemma-3n-E4B-it	424 / 1100	38.55%

RQ1 (Effectiveness). The models demonstrated moderate to high effectiveness depending on scale, with accuracy ranging from 38.55% to 67.36%, indicating that test smell detection remains a complex semantic classification task for current LLMs.

RQ2 (Per-category comparison). Test smells with clear syntactic markers, such as Exception Catching/Throwing, were consistently detected with high accuracy across models (up to 94% for Qwen-3-235B). In contrast, semantically complex smells such as Eager Test and Ignored Test proved most challenging, with accuracy below 35% for several models, indicating that these context-dependent patterns are significantly more challenging for LLMs.

RQ3 (Scale vs. performance). Within-family comparisons (GPT-OSS-120B vs. GPT-OSS-20B; Qwen-3-235B vs. Qwen-3.5-9B) revealed narrow accuracy gaps despite substantial differences in parameter count, suggesting that scaling alone yields diminishing returns for this task. The largest performance gap (28.81 pp) was observed between Gemma generations, indicating that generational improvements in training and architecture may have a stronger impact than parameter count. The MoE-based Qwen-3-235B outperformed the dense Qwen-3.5-9B by only 3.73 pp, suggesting that architecture alone is unlikely to explain substantial improvements.

4 Threats to Validity

Internal: Executing models with default provider parameters (temperature, top-p) may influence response determinism. Construct: We used only one prompt strategy (CRISPE with multiple choice), and the multiple-choice format restricts the response space, potentially introducing evaluation bias. External: Results are limited to Java; extending to other languages requires equivalent oracle tools. The Gold Set of 1,100 instances may be limited given the diversity of real-world code patterns.

5 Conclusion and Future Work

This paper presented NoseSense, an automated and reproducible benchmark for evaluating LLMs on test smell detection in Java unit tests. The empirical evaluation of six state-of-the-art models demonstrated that LLM effectiveness varies substantially across models and test smell categories. Although LLMs do not yet provide a reliable replacement for static analysis in this domain, the benchmark fulfills its primary objective: providing an extensible, reusable platform for tracking LLM capability evolution over time.

Future work will explore open-ended prompts, alternative prompting strategies (few-shot, chain-of-thought), extension to other programming languages (Python, Kotlin), expansion of the Gold Set, and investigation of the effect of explicit reasoning modes (thinking models) on test smell detection accuracy.

The NoseSense tool, dataset, and replication package are publicly available at: https://github.com/arieslab/nosesense.

Previous Work

This paper is a 2-page summary of the full paper published in the proceedings of the 11th Brazilian Symposium on Systematic and Automated Software Testing (SAST 2026) [3], co-located with CBSoft 2026. This work was developed under a PIBIC/CNPq scholarship and was also presented at the *Congresso UFBA 80 Anos* on July 8, 2026. Information about the institutional event is available at https://80anos.ufba.br/.

Artifact Availability

The NoseSense tool, including its source code, dataset, and replication package, is publicly available at: https://zenodo.org/records/20941814. The repository for the tool is available at: https://github.com/arieslab/nosesense.

References

[1] Gabriele Bavota, Abdallah Qusef, Rocco Oliveto, Andrea De Lucia, and Dave Binkley. 2015. Are test smells really harmful? an empirical study. *Empirical Software Engineering* 20, 4 (2015), 1052–1094.

[2] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A Survey on Evaluation of Large Language Models. *ACM Trans. Intell. Syst. Technol.* 15, 3, Article 39 (March 2024), 45 pages. doi:10.1145/3641289

[3] Magno Macedo, Lucas Lopes, Letícia Ribeiro, Tássio Virgínio, and Ivan Machado. 2026. NoseSense: Benchmarking LLMs for Test Smell Detection. In *11th Brazilian Symposium on Systematic and Automated Software Testing (SAST)*.

[4] Yuetian Mao, Junjie He, and Chunyang Chen. 2025. From Prompts to Templates: A Systematic Prompt Template Analysis for Real-world LLMapps. In *Proceedings of the ACM International Conference on the Foundations of Software Engineering (FSE)*. ACM. Industry Track.

[5] Seyed Mahmoud Sajjadi Mohammadabadi, Burak Cem Kara, Can Eyupoglu, Can Uzay, Mehmet Serkan Tosun, and Oktay Karakuş. 2025. A survey of large language models: evolution, architectures, adaptation, benchmarking, applications, challenges, and societal implications. *Electronics* 14, 18 (2025).

[6] Luciana Lourdes Silva, Janio Rosa da Silva, Joao Eduardo Montandon, Marcus Andrade, and Marco Tulio Valente. 2024. Detecting Code Smells using ChatGPT: Initial Insights. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (Barcelona, Spain) *(ESEM '24)*. Association for Computing Machinery, New York, NY, USA, 400–406. doi:10.1145/3674805.3690742

[7] Alejandro Velasco, Daniel Rodriguez-Cardenas, Luftar Rahman Alif, David N. Palacio, and Denys Poshyvanyk. 2025. How Propense Are Large Language Models at Producing Code Smells? A Benchmarking Study. In *2025 IEEE/ACM 47th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. 96–100. doi:10.1109/ICSE-NIER66352.2025.00025

[8] Tássio Virgínio, Luana Martins, Larissa Rocha, Railana Santana, Adriana Cruz, Heitor Costa, and Ivan Machado. 2020. JNose: Java Test Smell Detector. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering* (Natal, Brazil) *(SBES '20)*. Association for Computing Machinery, New York, NY, USA, 564–569. doi:10.1145/3422392.3422499