

Evaluating Code Smell Detection with Large Language Models

Saymon Souza

MSc Student in Computer Science | silvasouzasaymon@gmail.com

Goal

- Evaluate LLMs ability to identify code smells.
 - How do LLMs **perform** in detecting code smells?
 - What are the **strengths** and **weaknesses** of LLMs in detecting code smells?
 - How **consistent** are LLMs in detecting code smells **across different** software systems?

LLMs and Code Smells

- LLMs for code smell detection is key for three reasons:
 - Static analysis tools have shown **low performance in code smell detection** [1] and focus on **specific smells** [2].
 - Large Language Models (LLMs) have gained attention for addressing coding problems, but **their effectiveness in detecting code smells remains unclear**.
 - If LLMs can identify low-quality code, **they could also generate code free of these issues**.

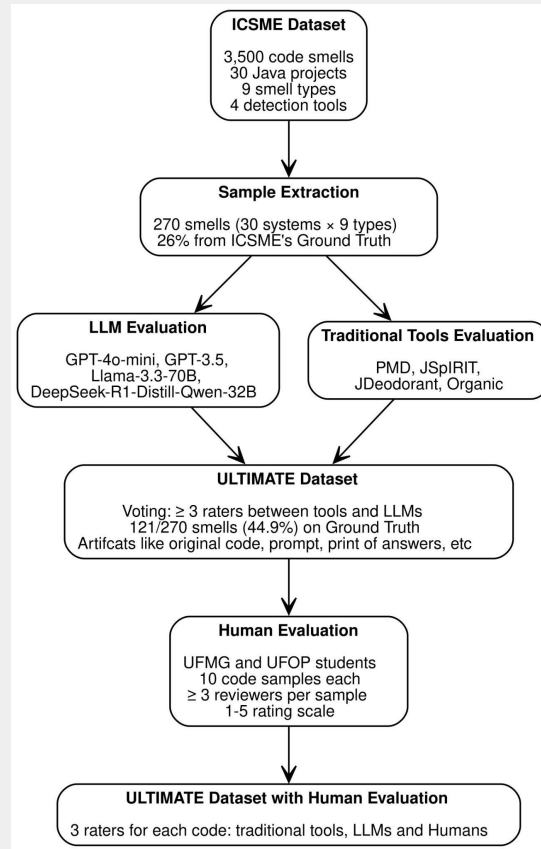
[1] Fernandes, Eduardo, et al. "A review-based comparative study of bad smell detection tools." Proceedings of the 20th international conference on evaluation and assessment in software engineering (ESEM). 2016.

[2] de Paulo Sobrinho, Elder Vicente, Andrea De Lucia, and Marcelo de Almeida Maia. "A systematic literature review on bad smells–5 w's: which, when, what, who, where." IEEE Transactions on Software Engineering 47.1 (2018): 17-66.

Research steps

- To answer the research goals, I divided my research in two steps:
 - Creation of a dataset of code smells analysed by traditional tools and LLMs;
 - Creation of a dataset with human analysis of the codes present in the first dataset.
- The plan is to use this research in (1) a paper we'll submit to TSE and (2) my dissertation.

Research steps



ULTIMATE Dataset

(Unified LLM Tool Inspecting MAnually Tailored Examples)

Original dataset

- We expand on Amanda et al. dataset from her paper on ICSME' 24.
 - Santana, Amanda, Eduardo Figueiredo, and Juliana Alves Pereira.
"Unraveling the Impact of Code Smell Agglomerations on Code Stability."
2024 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2024.

Original dataset

- Amanda et al. dataset includes over **3,500** code smells across **30 top-starred Java projects** on GitHub.
- The smells were evaluated by **four tools**: PMD, JSpirIT, JDeodorant, and Organic.
- It contains **nine different types** of code smells:
 - *Class-level*: Large Class, Data Class, Refused Bequest;
 - *Method-level*: Feature Envy, Intensive Coupling, Dispersed Coupling, Long Parameter List , Shotgun Surgery, Long Method.

Original dataset

- To add smells to their Ground Truth, they used a vote strategy in which two tools are used to detect the smell. If they agree that the instance has the smell, then it was added to their ground truth.

Smell	First Tool	Second Tool
Large Class	JDeodorant	JSplRIT
Refused Bequest	JSplRIT	Organic
Data Class	PMD	Organic
Long Method	JDeodorant	Organic
Feature Envy	JDeodorant	Organic

Smell	First Tool	Second Tool
Dispersed Coupling	Organic	JSplRIT
Intensive Coupling	Organic	JSplRIT
Shotgun Surgery	Organic	JSplRIT
Long Parameter List	PMD	Organic

ULTIMATE dataset

- We randomly extracted **270 smells** from the original dataset:
 - 30 systems
 - 1 smell from each of the 9 types
 - $30 * 9 = 270$
- 26% of the smells come from the original's dataset ground truth.
 - We chose to randomly select smells from inside and outside the ground truth to mitigate a bias regarding the presence of smells in the code.
 - Codes outside Amanda et al. Ground Truth might have smells detected by the LLMs that the tools couldn't agree with.

Chosen LLMs

- We ran a code smell detection on every code present in the dataset on these models:
 - From ChatGPT, models *gpt-3.5-turbo* and *gpt-4o-mini*;
 - From Meta, model *Llama-3.3-70B-Instruct*
 - From deepseek-ai, model *DeepSeek-R1-Distill-Qwen-32B*
 - To run models *DeepSeek-R1-Distill-Qwen-32B* and *Llama-3.3-70B-Instruct* we used *Hugging Face*.
- We chose these models because of both their easy to use nature and research interest, given that they are some of the most common models present in recent LLM papers.
- We aimed for deterministic, predictable outputs. So the temperature was set to **0** for all four models.

Prompt

- We chose a prompt existent in the literature and presented in a strong event [4]
- It relies on a *zero-shot* technique;
 - Which means that the LLMs were asked to detect code smells without having seen any examples beforehand.

[4] Silva, Luciana Lourdes, et al. "Detecting code smells using chatgpt: Initial insights." Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). 2024.

Prompt

The list below presents common code smells (aka bad smells). I need to check if the Java code provided at the end of the input contains at least one of them.

- * **Large Class**
- * **Data Class**
- * **Refused Bequest**
- * **Feature Envy**
- * **Intensive Coupling**
- * **Dispersed Coupling**
- * **Long Parameter List**
- * **Shotgun Surgery**
- * **Long Method**

Could you please identify which smells occur in the following code? However, do not describe the smells, just list them.

Please start your answer with "YES I found bad smells" when you find any bad smell.

Otherwise, start your answer with "NO, I did not find any bad smell".

When you start to list the detected bad smells, always put in your answer "the bad smells are:" amongst the text your answer and always separate it in **this format: 1. Long method, 2.**

Feature envy ...

[Java source code with the smell]

Output

















- Since all models received the same prompt, all outputs followed the same format.

```
YES, I found bad smells. The bad smells are: 1. Long  
Parameter List 2. Data Class
```

Generated Artifacts

- We sent all class codes to the LLMs and collected its answers.
 - Although some smells are in the method level, we decided to use the full class in the LLMs evaluations.
 - This was made because the context outside the method is important for some smells like Feature Envy and Shotgun Surgery.
- For each sample we saved the original code, the prompt used and a print of the LLM answer.

Generated Artifacts

Nome	
 DBeaverCore.java 	
 organicDiCo.csv 	
 DiCo_DBeaverCore.py 	
 DiCo_DBeaverCore.png 	
 DiCo_gpt-4o-mini_DBeaverCore.py 	
 DiCo_gpt-4o-mini_DBeaverCore.png 	
 DiCo_Llama-3.3-70B-Instruct.png 	
 DiCo_DeepSeek-R1-Distill-Qwen-32B.png 	

Ground Truth

- Each one of the 270 codes present in the dataset were evaluated by 6 raters:
 - Two traditional tools and four LLMs.
- We used a similar voting system as the one present in Amanda at el. dataset
 - We discarded *gpt-3.5-turbo* from the voting system to not give ChatGPT two votes. Choosing to keep the more modern model.
 - Each code had a smell attached to coming from the original dataset.
 - For that code with a smell to be added in our ground truth it needed 3 raters to detect the smell present in the original dataset.

Ground Truth

Class	Code Smell	JDeodorant	JSpirit	Organic	PMD	gpt-4o-mini	Llama-3.3-70B	DeepSeek-R1-Distill-Qwen-32B	Oracle TSE
MethodInfo	Data Class			Yes	Yes	No	Yes	Yes	Yes
BitMatrix	Shotgun Surgery		Yes	No		No	No	No	No
RetrofitError	Long Parameter List			Yes	No	Yes	Yes	Yes	Yes

Table 2: Example of the votation system

Ground Truth

- Out of the 270 code smells, 121 are present in our ground truth.
 - This represents 44,9% of our dataset.
 - In our ground truth, 26,7% of codes are also present in Amanda et. al dataset ground truth.
 - This was a reason we randomly selected codes from Amanda et. al dataset.
 - The LLMs won't always agree with the tools, which opens space for a human evaluation to understand if developers of different experiences agree with our ground truth.

Human Evaluation

Goals

- With the help of undergrad computer science students from UFMG and UFOP, we want to:
 - Evaluate the quality of our ground truth;
 - Investigate whether humans agree with the LLMs in identifying code smells.

Setup

- We divided our experiment in two parts:
 - Phase 1 with the students of the course of Software Engineering of UFMG;
 - Phase 2 with the students of the course of Software Engineering of UFOP.

Setup

- The human evaluators will rate all codes in our dataset.
 - UFMG students evaluated 188 code samples - which represents 69% of our dataset .
 - UFOP students will evaluate the rest, 82 code samples - which represents the other 31% of our dataset.

Evaluations

- Each student evaluated **10** code samples.
 - Our goal was to ensure that all code samples are reviewed by at least 3 different students.
 - Given that each experiment would have a maximum of 100 minutes, this allowed the students to take their time during the evaluations.

Evaluations

- To improve the quality of the data collected from students, two out of the ten evaluations were designed to **help identify discrepancies**.
 - For example, we might assign a code to a student where the LLMs identified the Data Class smell.
 - However, instead of providing a form specific to Data Class, we will present a form for a different smell **found by one of the LLMs**.
 - This allows us to verify whether the detection was a discrepancy.

Codes and students allocation

- All code samples were randomly distributed among the students enrolled in the class.
 - The distribution was designed to prevent students evaluating the same code more than once.
- Students received a link to a Google Sheets document containing a tab labeled with their Student ID.
 - Each student found the code samples assigned to them in their respective tab.

Presentation for the students

- We began the experiment with a short presentation to provide the basic knowledge students need to complete the evaluations.
 - The presentation covered key concepts and gave an overview of the nine types of code smells included in the experiment.
 - For all type of smells, code examples will also be provided;
 - Each presentation lasted between 20 and 30 minutes.

Forms

- Students were asked to complete three forms, in the following order:
 - **Consent:** Standard consent form where participants gave informed consent to participate in the experiment.
 - **Background:** This form collected information about the students' professional experience and their knowledge of Java and code smells.
 - **Evaluation:** This is where students submitted their code evaluations. Each one of the nine smells has a specific form.
- All forms were provided through Google Forms.
 - The first question asked for the student's name and Student ID to allow for identification.

Background Form

- We asked students three questions about their professional experience:
 - Which Software Engineering topics they are familiar with, selected from a predefined list;
 - How many years of professional experience they have;
 - How they would rate their skill level in Java and code smell detection, using a scale from *Never heard of it* to *I'm an expert*.

Evaluation Form

- In this form, students provided their evaluations of the code samples assigned to them. They answered the following:
 - Which class they evaluated;
 - How strongly they believe there is a code smell in the class, using a scale from 1 to 5, where 1 means *Definitely no code smell* and 5 means *There is definitely a code smell*;
 - A justification for their answer, indicating which part(s) of the code influenced their decision.

Experiment Pilot

- We piloted the experiment with LabSoft students and collaborators for two weeks.
 - Fortunately, only one major problem was found and solved within hours after the start of the test.
 - Most of the feedbacks were related to improvements to the quality of life of the participants.

Experiment

- Phase 1 of our experiment was on May 15th with students of UFMG.
 - We had 50 students during the experiment.
- Phase 2 of our experiment will happen on June 4th with students of UFOP.
 - We expect around 35 students to be present at the day of the experiment.

Collected data so far

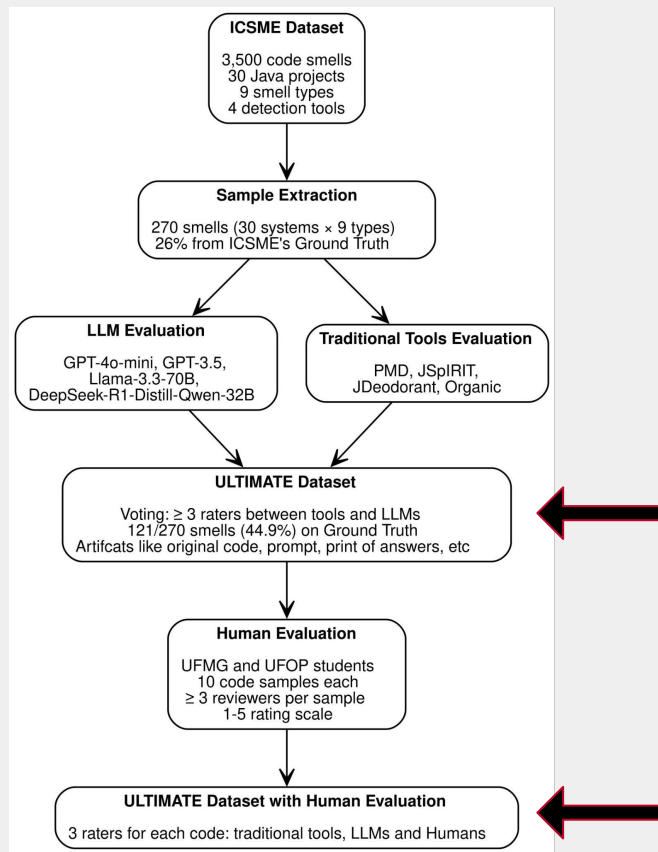
- We received 577 from 50 UFMG students + LabSoft students and partners!
 - From the 188 codes in the experiment, 181 received at least 1 eval.
 - 156 codes received at least two evals.
 - 99 codes received at least three evals
 - 6 codes received more the three evals
- We also received 132 discrepancy evals
 - 30 codes received two discrepancy evals

Next Steps

- Finish second phase of our experiment
- Crunch the numbers and make the analysis portion of the TSE paper
- Adjust the text to fit a dissertation

Contributions

Contributions



Future Work

Future Work

- Automation of the code smell detection process to expand the dataset.
- Expand our LLMs library adding especially ones with focus on coding (Sonnet, Claude, Copilot, etc)
- Now that we have one of our datasets fully evaluated by humans, we can use the best examples to train LLMs in order to apply few-shot learning on codes.
 - This could improve detection results from the LLMs;
 - Get us closer to the goal of determining if LLMs are reliable source of code smells detection.

Evaluating Code Smell Detection with Large Language Models

Saymon Souza

MSc Student in Computer Science | silvasouzasaymon@gmail.com