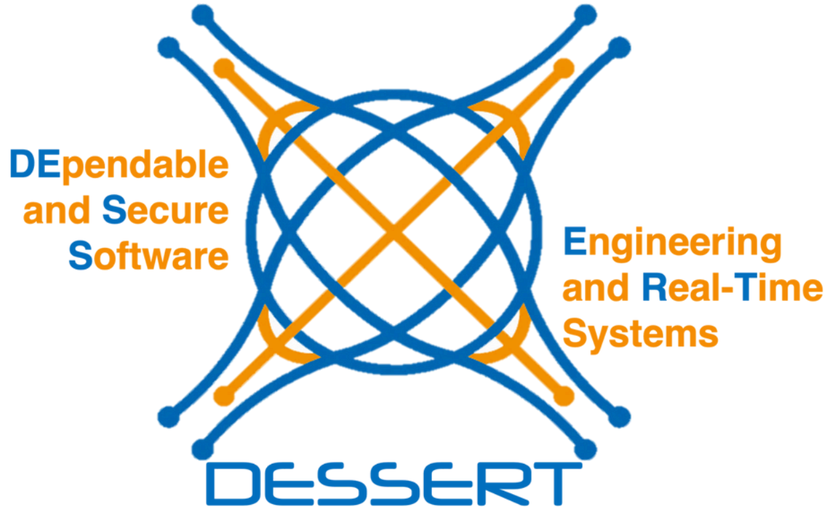
Discovering and classifyng residual defects in the new era of AI-based Code Generators

Domenico Cotroneo,
Pietro Liguori, Giuseppe De Rosa, Benedetta Gaia Varriale

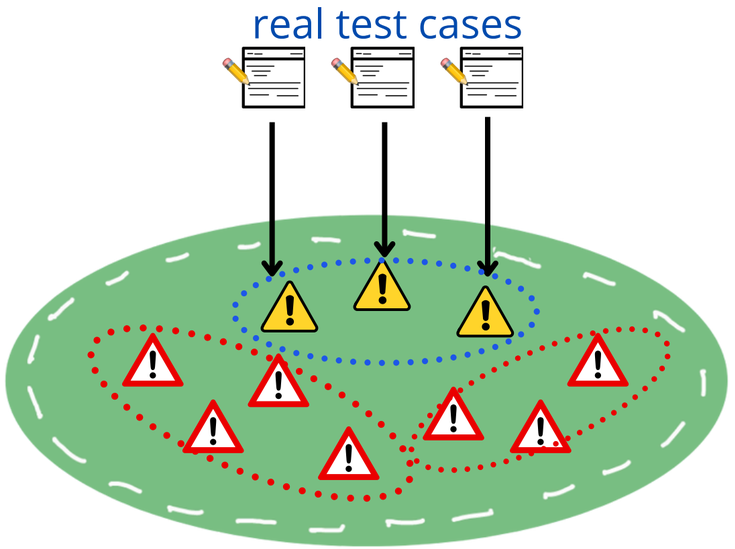


https://dessert.unina.it

The Challenge of Predicting Residual Faults

Residual faults are bugs that remain after extensive testing.

It is still an open and challenging issues



AI Code Generators: A New Source of Residual Bugs

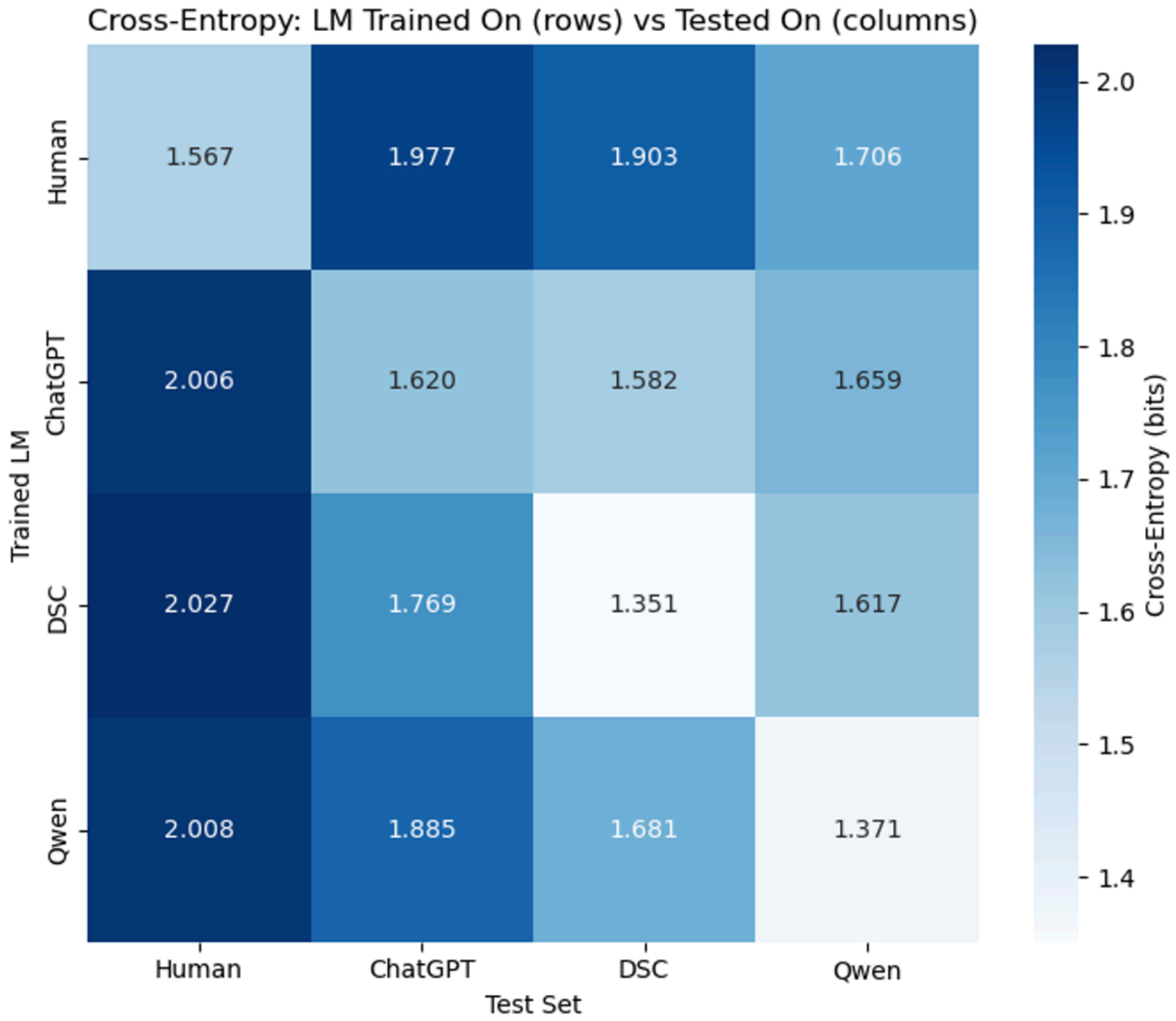
- AI-based code generators (e.g., Copilot, ChatGPT) produce large volumes of code.
- AI code generators aren’t only used to write new features. They’re increasingly integrated into comemercial tools that apply patches automatically
- (e.g., CodeQL, CoPilot, sourcery etc...)
- AI-generated code is syntactically correct, but often buggy, vulnerable, and unnatural.
- Subtle, non-obvious (maybe unknown and/or difficult to detect) bugs can escape traditional review and testing.

Write a Java Funtion that multiplies two matrixes that stores the result in a static cache

```
import java.util.*;

public class LeakyMatrixMultiplier {
  // Simulate a cache that is never cleared (memory leak)
  private static Map<String, int[][]> resultCache = new HashMap<>();

  public static int[][] multiply(int[][] A, int[][] B) {
    int rowsA = A.length;
    int colsA = A[0].length;
    int colsB = B[0].length;

    int[][] result = new int[rowsA][colsB];

    for (int i = 0; i < rowsA; i++) {
      for (int j = 0; j < colsB; j++) {
        for (int k = 0; k < colsA; k++) {
          result[i][j] += A[i][k] * B[k][j];
        }
      }
    }

    // Store result in a static cache that never release entries
    String key = UUID.randomUUID().toString();
    resultCache.put(key, result);

    return result;
  }
```

The use of statistical metrics

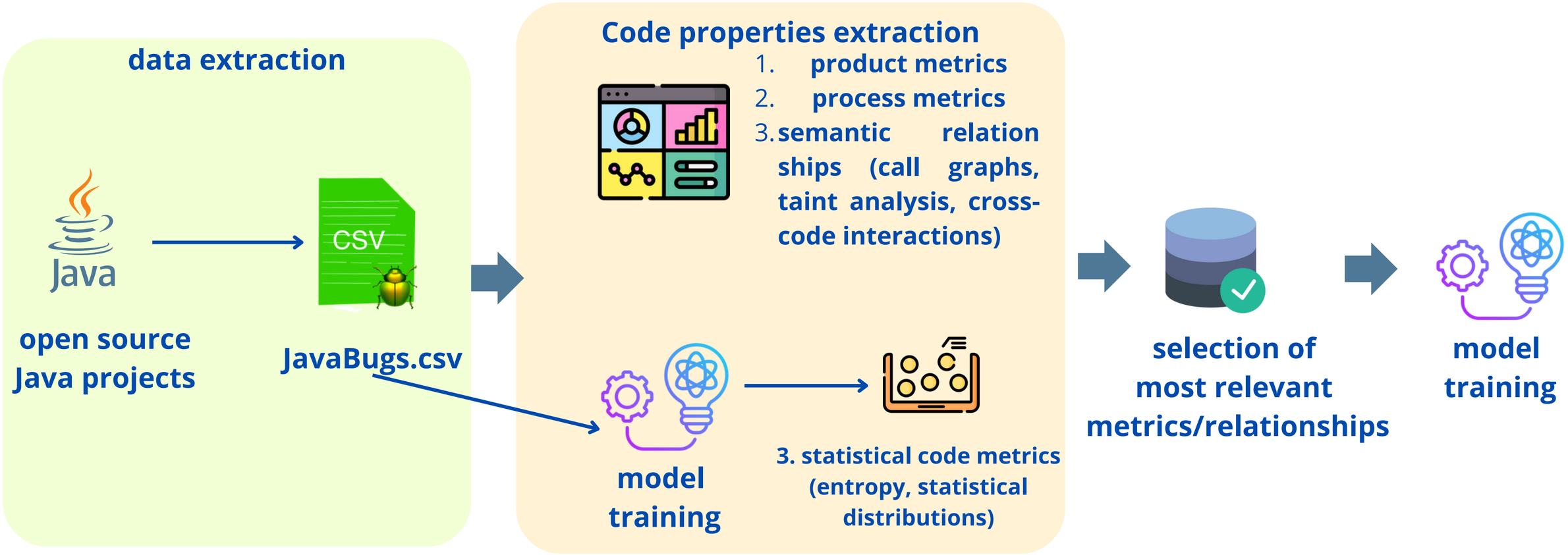
- Process metrics tell us how the code changes over time.
- Product metrics tell us what the structure and complexity are.
- But neither tells us how unusual or unnatural the code is.
- Statistical metrics e.g., cross-entropy(which captures the naturalness of code, i.e, its ability to be repetitive and predictable) are beign used in literature



507,000 code samples (Python and Java). We used ChatGPT, DeepSeek-Coder, and Qwen-Coder,

Our idea that targets Java systems

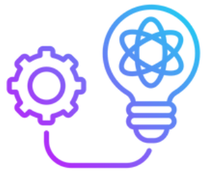
We started with more thatn 200K Java funtions form human-made code repositories
Make classification through software code metrics



contributions (ongoing work)

JavaBugs, a new dataset with (residual) faulty methods and classes from open source Java projects. It provides complete information about the source of the code

Subset of relevant metrics and reltionships that enable accurate residual fault prediction



Model for classification of residual bugs through the relevant metrics

preliminary experiments - metrics extraction

- **Metrics extraction:**
  - we want to study how the combination of three different types of metrics could be useful in residual faults prediction. We considered:
    - **process metrics**
    - **product metrics**
    - **statistical code metrics (entropy,** which captures the naturalness of code, i.e, its ability to be repetitive and predictable, as it is the product of human effort (Hindle et al.))

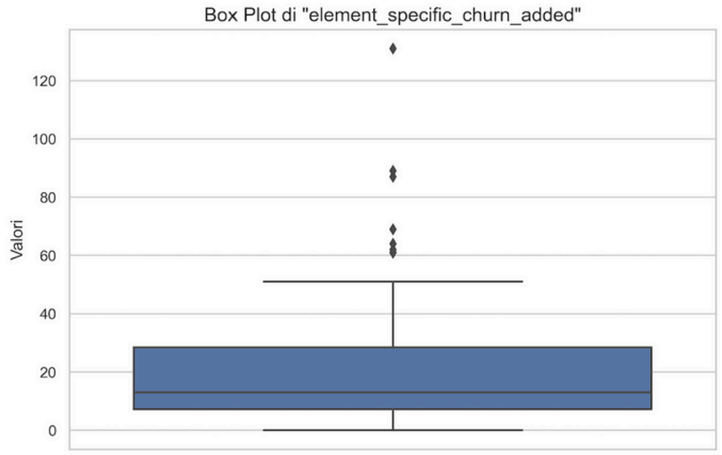
preliminary experiments - entropy

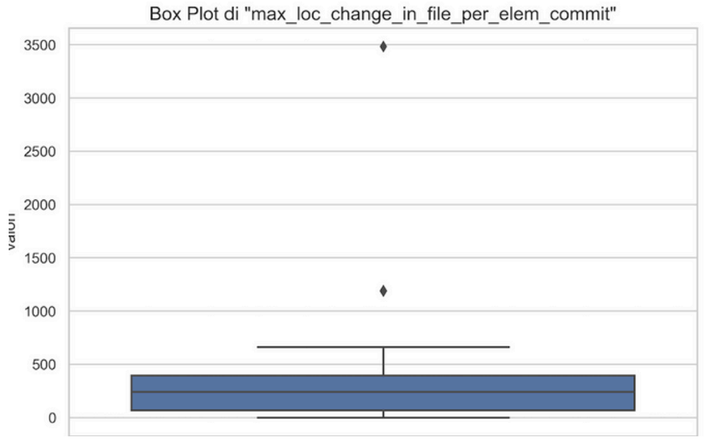
- We observed a difference at line level:
  - the entropy of a fixed or faulty line is higher than the one of correct line,

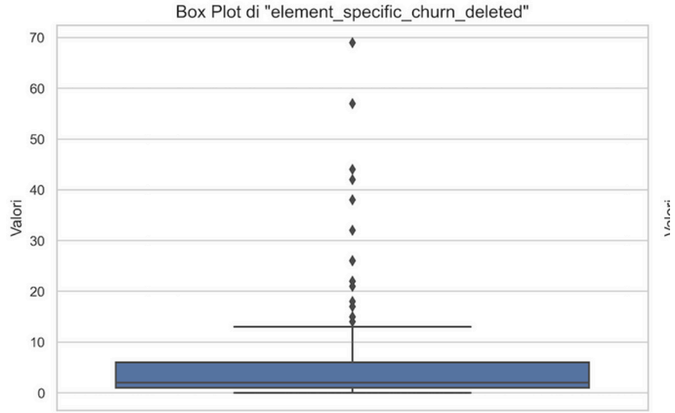
- This leads us to the conclusion that also **the fixed lines are less natural than the correct ones**(Jiang *et al.*) In the next steps we will understand how to use this information.

preliminary experiments - process metrics

- In this preliminary phase, we picked these three metrics because their boxplots show wide spreads and clear outliers, making it easy to spot elements with unusually high new code additions, overall churn from commits, and single-commit change spikes







Ongoing work and conclusion

- Preliminary experiments suggested use that
  - **we can use cross-entropy even though we have to shift the analysis from function level to line level**
  - Process metrics can be also used to leverage the characterization
  -
- We are train our final model and analse the results

- We are extending the dataset that contains both AI- and human -generated code in order to compare the types and the density of residual faults