ML4SE Research between academia and industry

Egor Bogomolov
ML Research Lead at JetBrains

Contents

- ➔ Getting to know each other
- ➔ Development of software development
- ➔ Research in SE industry
- ➔ Obstacles on the edge of academia and industry (and how to avoid them)

Let’s get to know each other

JetBrains Research

ML Division:

- ➔ Code Modeling
- ➔ Code Editing
- ➔ AI Agents & Planning
- ➔ Federated Compute
- ➔ Human-AI Experience (HAX)

Applied Division:

- ➔ Software Testing
- ➔ Code Comprehension
- ➔ Debugging
- ➔ Automated Program Repair
- ➔ Dynamic Program Analysis

\+ **Education Research**
\+ **Collaborations**

JetBrains Research

ML Division:

- ➔ Code Modeling
- ➔ Code Editing
- ➔ AI Agents & Planning
- ➔ Federated Compute
- ➔ Human-AI Experience (HAX)

Applied Division:

- ➔ Software Testing
- ➔ Code Comprehension
- ➔ Debugging
- ➔ Automated Program Repair
- ➔ Dynamic Program Analysis

\+ **Education Research**

\+ **Collaborations**

I'm responsible for this part but happy to talk about everything 🤝

Machine learning for SE (ML4SE)

- ➔ **Writing code**: auto completion, synthesis, search
- ➔ **Debugging**: finding errors, bug fixing, advanced static analysis
- ➔ **Enhancing structure and code quality**: generation of comments, commit descriptions, refactoring recommendation, suggestion of identifier names
- ➔ **Maintenance and support**: finding duplicate issues, bug triage
- ➔ ...

Specifics of ML4SE

- ➔ Source code is **not only text**
- ➔ Available data is **not only source code**
- ➔ Solutions should be **integrated** into everyday tools
- ➔ Software developers are very **picky** in their tools
- ➔ It looks like we have **lots of data**, but it's **not always** the case

What’s cool in SE recently?

GitHub Copilot (not recently already)

Coding agents everywhere

Next Edit Suggestions

```
public class UserProfileManager {
    public void updateProfile(int userId) {
        String userPhone = fetchUserPhone(userId);

        displayUserPhone(userPhone);
        validateUserPhone(userPhone);
        logPhoneNumber(userPhone);
        saveUserPhone(userId, userPhone);
        notifyUserUpdate(userId, userPhone);
    }
```

ML turned into AI
and became a foundation for many tools
for software developers

But where most of it originated?

In research!

A few works that led to this (of many and many)

- ➔ On the Naturalness of Software
- ➔ Evaluating Large Language Models Trained on Code
- ➔ Can Language Models Resolve Real-World GitHub Issues?
- ➔ Learning to Represent Edits
- ➔ Fast Inference from Transformers via Speculative Decoding

Some works are extremely impactful, even though yet to revolutionize industry

How research looks in SE industry

What research means in SE industry

➔ **Goal**: reduce uncertainty for future development tools

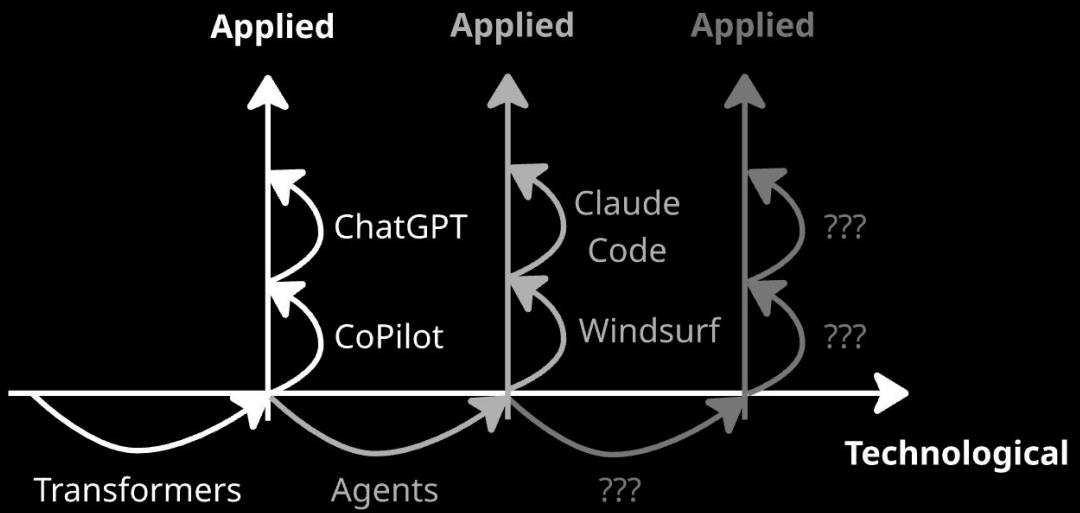
➔ **Outputs**: methods, prototypes, datasets, evaluation tools, design docs, papers

➔ **Measured by**: knowledge created & unlocked possibilities

➔ **Time horizon**: weeks → months → years (portfolio, not a single bet)

Technological and Applied

- **Technological research**: extend the limits of what's possible with technologies
- **Applied research**: find the best ways to solve user or product problems end-to-end



SE companies need both

- ➔ Development of **new technologies**
- ➔ Tailoring them to **practical tasks**
- ➔ Turning into **product features**
- ➔ Find **new constraints** and repeat
  - ✓ latency, memory, privacy, UX, ...

Research has huge impact on the advancements in SE tools

Let’s check when it does not go as smooth

Any practical setting is extremely specific

Research work should generalize to it

When is generalization failing?

➔ Results are **hard to reproduce**

➔ **Overfitting** to the benchmark

➔ Missing or **unrealistic constraints**

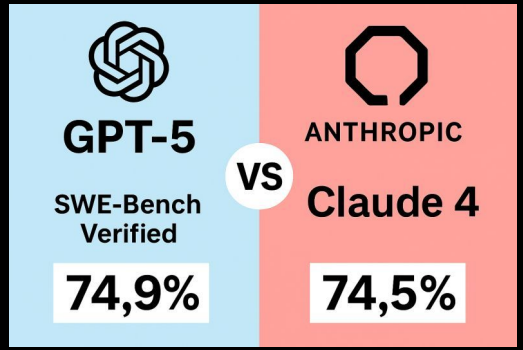
➔ **Evaluation mismatch** in metrics or datasets

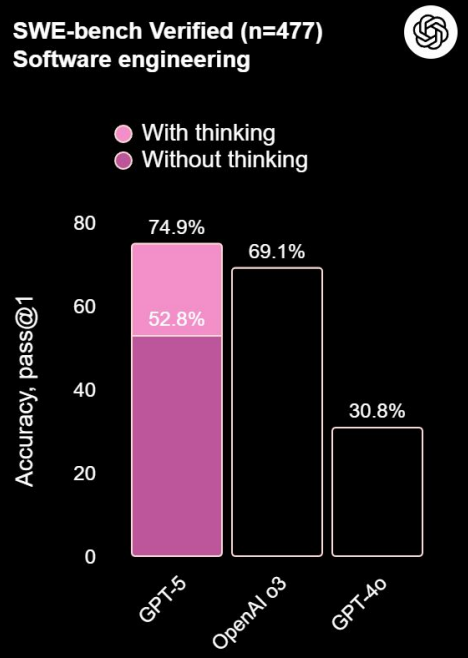
Case #1: reproduction

➔ Datasets or scripts are not published

➔ Paper and code diverge

➔ Undocumented modifications to benchmarks





How to make reproduction easier?

- Publish artifacts
- Document your artifacts and decisions

Case #2: overfitting

➔ Many small modifications that result into a marginal improvement

➔ Competition-like squeezing of results

➔ Reward hacking, methods that exploit benchmark imperfections

[The SWE-Bench Illusion: When State-of-the-Art LLMs Remember Instead of Reason]

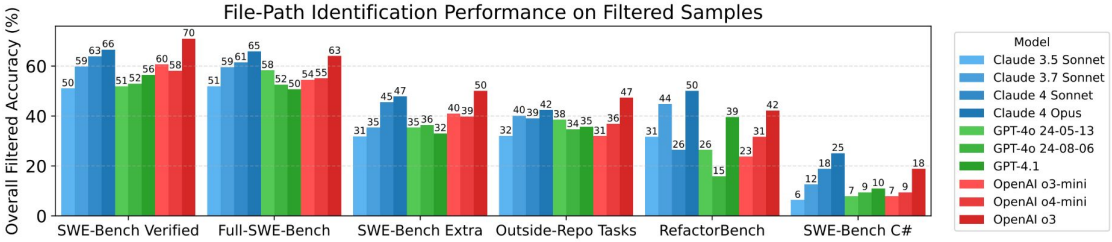


Figure 4: Comparison of models' filtered accuracy on the file path identification task across benchmarks.

How to avoid overfitting?

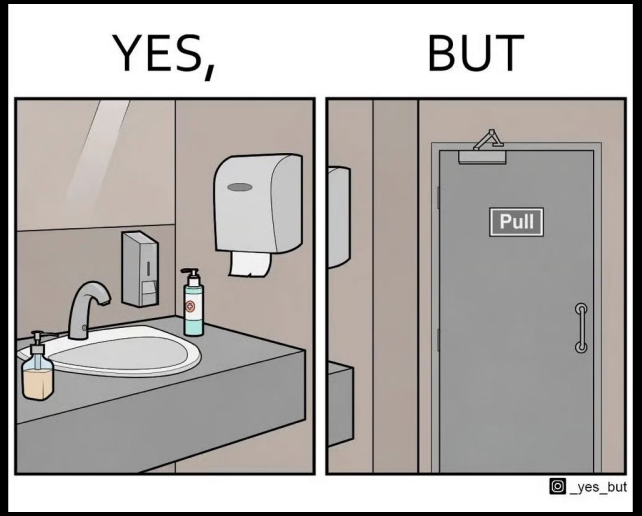
- Use diverse benchmarks
- Test hypotheses rather than squeeze quality
- Statistical testing, ablation studies

Case #3: unrealistic constraints

➔ Model improves code completion quality...
...at the cost of 100x slower inference

➔ Solution can find where to move a method...
...but needs an LLM call for each project file

➔ Approach generates amazing tests for a class...
...but requires 1,000s runs of the entire test suite



Why think about constraints?

- Get to know more about the methods
- New ideas for research!
- Making impact with engineering

Case #4: evaluation mismatch

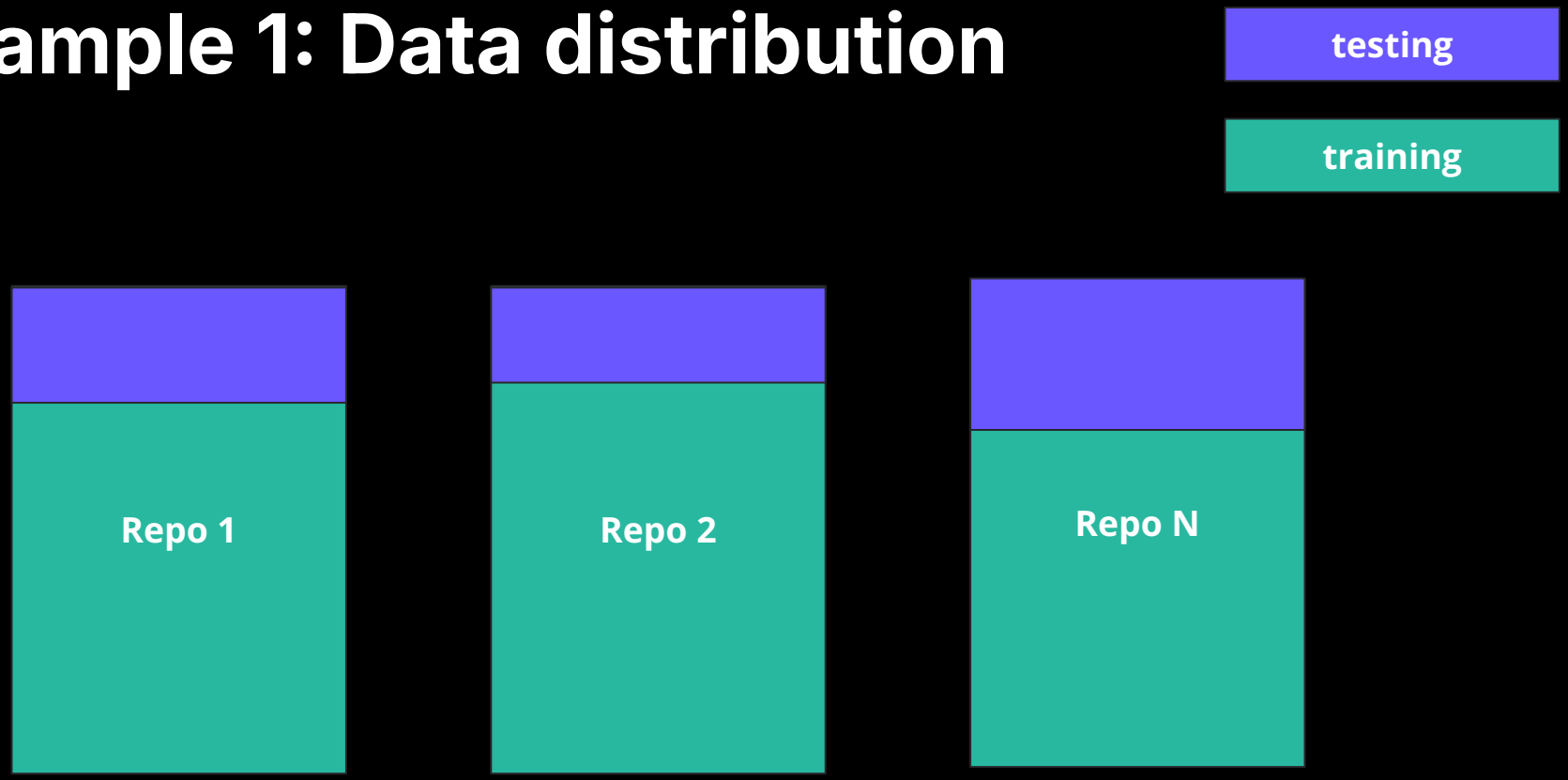
We evaluate to assess solution generalizability beyond the training data

Example 1: Data distribution

testing

training

Example 1: Data distribution

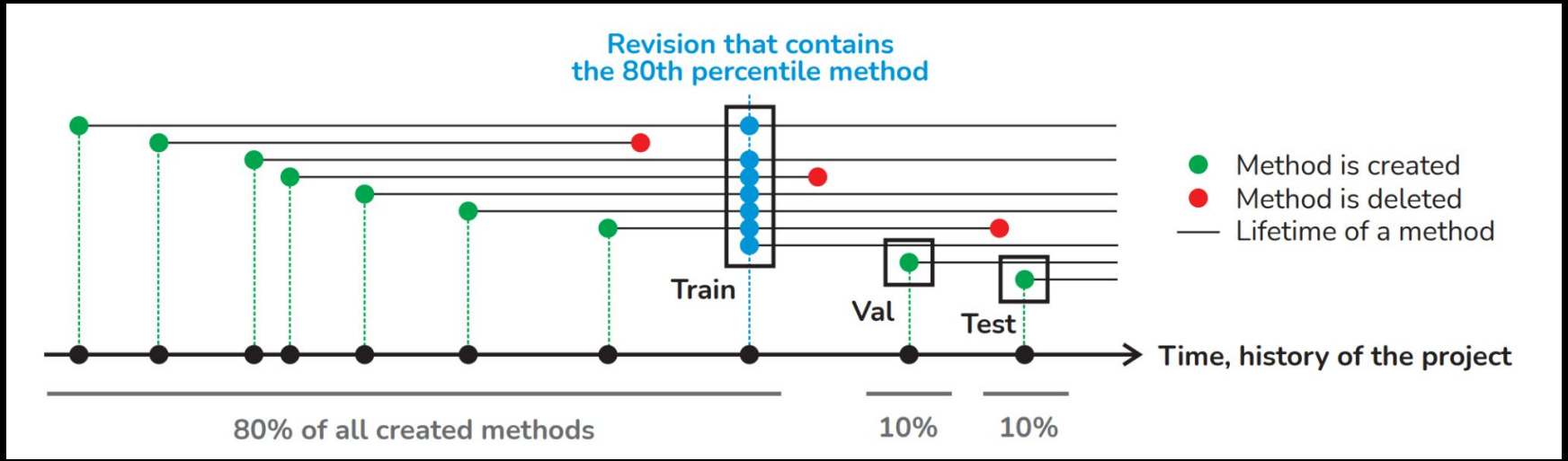


Example 1: Data distribution

The dataset should reflect the data distribution of your ultimate task

What else can go wrong?

Example 2: Data leakages from the past



arxiv.org/abs/2206.03333

Example 2: Data leakages from the past

Model	Training data, F1	Testing data, F1
CodeTransformer	48.5	40.9
Code2Seq	47.1	34.8
TreeLSTM	38.5	26.9

Model has not seen this data before

“training” occurs earlier than the “testing”

arxiv.org/abs/2206.03333

Example 3: Proxies that you use

Authorship attribution: Motivation

To attribute malware

To fix inaccurate or missing authorship information in software projects

To solve the clone detection task directly

Authorship attribution: Utilized datasets

Submissions from programming competitions such as Google Code Jam

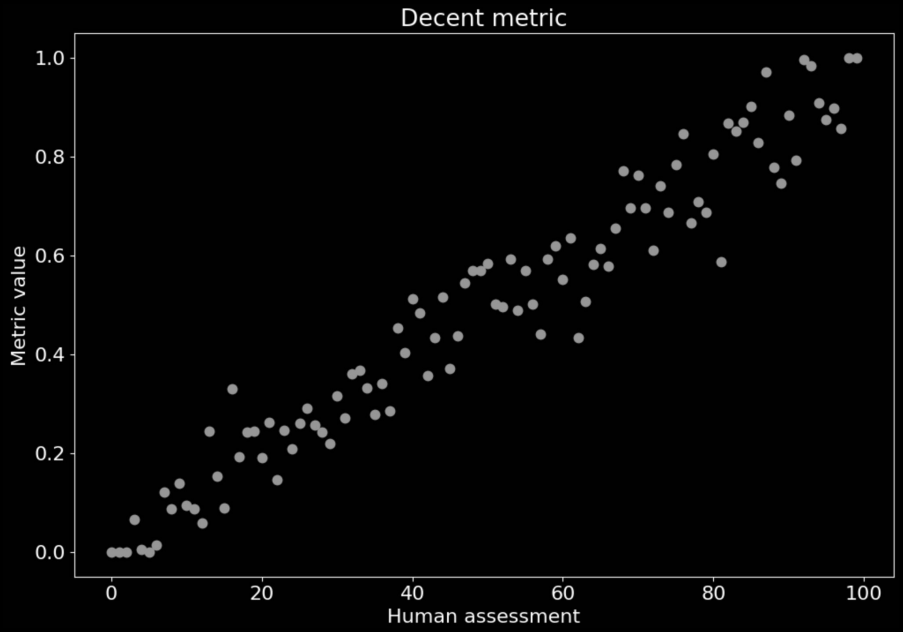
Repositories from GitHub, single author each

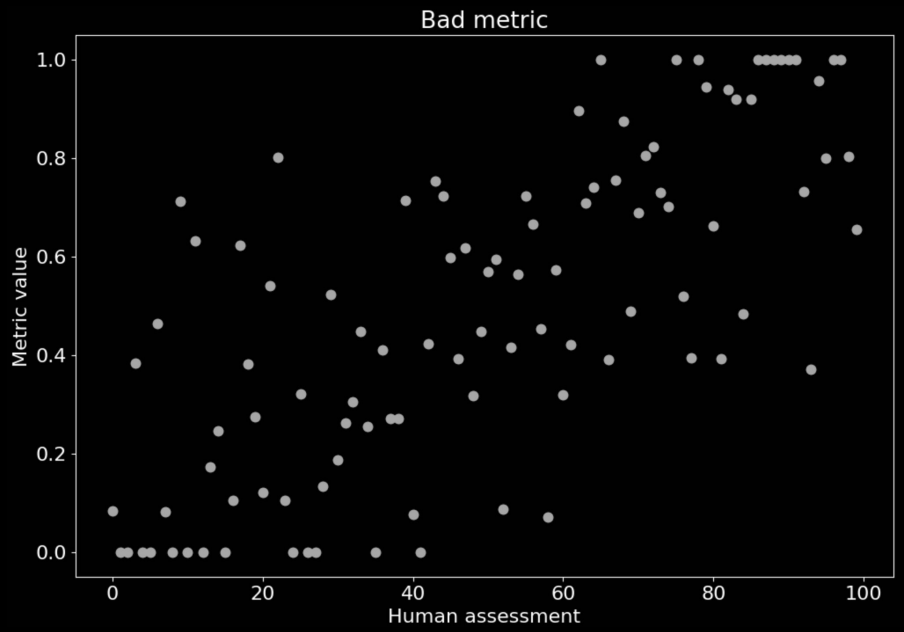
Students' assignments

arxiv.org/abs/2001.11593

What else do you need for evaluation?

Example 4: Metrics can be tricky





Case #4: evaluation mismatch

- ➔ **Align used data** with the desired use cases
- ➔ Pay attention to **metric selection**
- ➔ Test for **statistical significance**
- ➔ Benchmark should be **well correlated** with the target

Should all of it bother you? Not necessarily

But it may open new directions for research

Instead of conclusion, cool work

Conventionally: “Working on new architectures and getting SOTA” 💸

Hopefully, after this talk:

➔ Working on high-quality data, evaluation setups, metrics – very cool 👍

➔ Improving over axis other than quality – very cool 👍

➔ Going an extra mile to prepare well-documented artifacts – very cool 👍

To learn more and collaborate:

JETBRAINS