WHY DO YOU FAIL ME MR. BOT?

2025/10/28

Amaan Ahmed, Asim Waheed, Youssef Souati

CS846 – Project Presentation

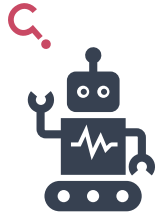
Analyzing Pull-Request Failures in the AIDev Dataset

AI Coding Agents now author hundreds of thousands of PRs

But many PRs fail to be merged or are reverted

Our goal:

Understanding *why*



Why study PR failures?

PR acceptance = natural test of trust in AI teammates

Prior work[1] showed agent PRs accepted far less than humans

Recent data suggests trend reversed

Understanding failures → Improving human-AI workflows

[1] Wyrich, M., Ghit, R., Haller, T., & Müller, C. (2021). Bots Don't Mind Waiting, Do They? Comparing the Interaction With Automatically and Manually Created Pull Requests. *2021 IEEE/ACM Third International Workshop on Bots in Software Engineering (BotSE)*, 6–10. https://doi.org/10.1109/BotSE52550.2021.00009

The AIDev Dataset

- 900,000+ agent-authored PRs across 100,000+ repos
- Metadata on PRs, reviews, comments and timelines
- Rich comparisons between agents and humans
- Useful to study PR failure across authors and contexts

Research Questions

RQ1: Are agent-authored PRs more likely to succeed than human-authored PRs?

RQ2: What factors lead to rejection of agentic PRs?

RQ3: What practices make agentic PRs more successful?

METHODOLOGY

What even is a “failed” PR?

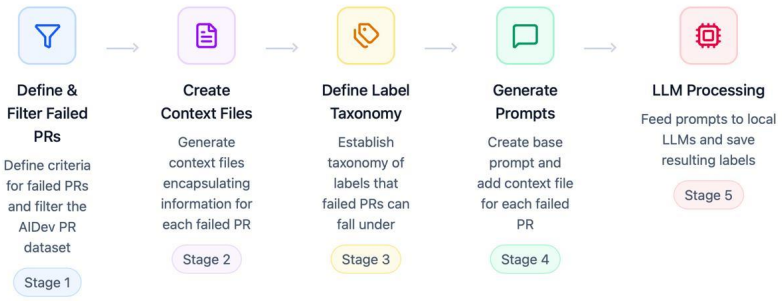
Dataset Details

- Datasets:
  - Curated AIDev Dataset: 33,596 PRs.
- Definition of failed PRs.
  - PR state is closed and *merged_at* is NULL.
  - PR state is open but no activity for 180 days.
- 7,646 failed PRs comprising 7,270 closed-but-unmerged and 376 stale.

Research Question 1

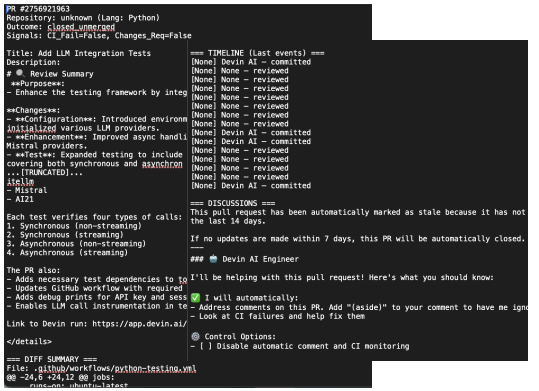
- Extract 6,618 human PRs from curated dataset
- Apply same failure rule → 1,204 failed PRs.
- Compare task-type distributions and failure rates by task type.

Research Question 2



RQ2: Stage 2

- PR context files include:
  - Repository metadata
  - Diff summary
  - Event timeline
  - CI/workflow signals
  - Review text
  - Commit messages
  - Final PR outcome



RQ2: Stage 3 (Failure Taxonomy)

Group	Description	Category
TECHNICAL (T)	Code, tests, style, architecture	T1: CI or Test Failure T2: Code Quality or Correctness
FIT_OR_VALUE (F)	Feature/requirement mismatch, not needed, wrong direction	F1: Wrong feature, not needed, or misaligned
PROCESS_OR_OWNERSHIP (P)	Time, priority, unresponsiveness, social/review issues, policy	P1: Low priority P2: Author Unresponsive P3: Review/social/coordination conflict P4: Policy/compliance/bureaucracy
REDUNDANCY_OR_OBSOLETE (R)	Duplicate feature or superseded work	R1: Duplicate feature R2: Obsolete PR
INVALID_OR_SPAM (S)	Spam, noise, non-meaningful PRs	S1: Invalid/Spam/Nonsensical
UNKNOWN (U)	Not enough information	U1: Unknown

Table 1: Failure taxonomy used for classifying rejected PRs.

```
{
  "id": "T1_CI_OR_TEST_FAILURE",
  "group": "TECHNICAL",
  "definition": "The PR is not merged mainly
      because tests, builds, or CI checks fail due
       to issues introduced or exposed by the PR,
      and the author never fixes them.",
  "typical_signals": [
    "ci_failed = true",
    "Timeline events like 'ci_failure', '
        workflow_run.completed: failure'",
    "Review comments: 'tests are failing', 'please
         fix CI', 'build is broken'",
    "No later commit that fixes the failures
        before close"
  ]
}
```


RQ2: Stage 4 and 5

- Base prompt + context file = prompt for each failed PR
- Output: (label, justification, confidence)
- Output JSON-enforced
- LLaMa-3.2-3b and Qwen2.5 used

```
You are an assistant that classifies why GitHub pull requests failed.

A "failed" pull request in this task is one that was CLOSED WITHOUT being
merged, or has remained OPEN and STALE for a long time.

Your goal is to read the pull request context and decide the SINGLE most likely
primary reason why it did not get merged, according to the taxonomy below.

If there is genuinely not enough information in the context to infer a reason,
you MUST use the UNKNOWN label (U1.UNKNOWN).
Do not guess when the evidence is missing or extremely weak.

Failure Taxonomy: <TAXONOMY_BLOCK>
TASK:
1. Read the PR context carefully.
2.  Decide the SINGLE most likely primary reason_label from the taxonomy
above.
3. Write a short reason_summary (1–2 sentences) explaining why you chose that
label.
4. Estimate a numeric confidence between 0 and 1 for how sure you are.

IMPORTANT:
- Choose exactly ONE reason_label.
- Do NOT invent details that are not supported by the context.
- If the signals are mixed, choose the reason that best explains why the PR
ultimately failed.
- If there is not enough information, use U1.UNKNOWN.

OUTPUT FORMAT (JSON ONLY):
You MUST output exactly one JSON object with this schema:

{
  "pr_id": <integer>,
  "reason_label": "<taxonomy label>",
  "reason_summary": "<1-2 sentence explanation>",
  "confidence": <float between 0 and 1>
}

All string values (reason_label and reason summary) MUST be enclosed in
double quotes to form valid JSON.

- pr_id MUST match the pr_id given below.
- Do NOT include any extra keys.
- Do NOT include any text before or after the JSON.

Now read the PR context below and produce the JSON output.

PR ID: <pr_id>
PR Context: <pr_context>
```

Research Question 3

- Combine the following data sources:
  - Failure reasons from RQ2
  - PR-level metadata (PR size, number of reviewers, etc.)
  - Repo-level metadata (Age, stars, etc.)
- Analyze failure reasons grouped by each variable
  - Report variables with significant differences for different values

RESULTS

So, why DO you fail me, Mr. Bot?

Recap

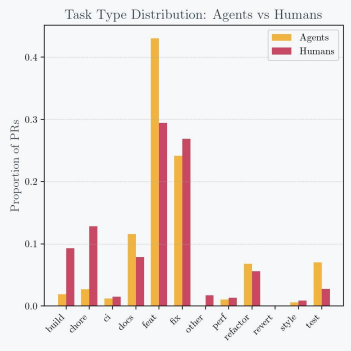
- Questions to answer:
  - Do agents fail more than humans?
  - Why do agent PRs fail?
  - What practices and contexts make agents more likely to succeed?
- All numbers here from curated dataset:
  - ~33K agent PRs.
  - ~6.6K human PRs.
  - Popular, actively maintained repos.

RQ1 – Do Agents Fail More Often Than Humans?

- Using curated repos:
  - Agents: 22.79% failure rate
  - Humans: 19.70% failure rate
- Using full AIDev dataset agents perform better (8.18% failure)
  - Due to less popular repos and less scrutiny during review.

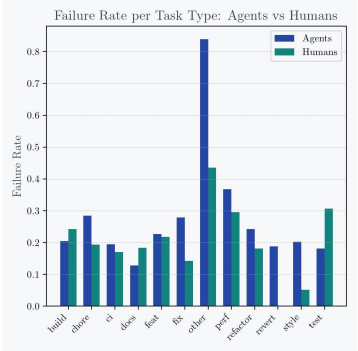
In serious, high-activity projects, agents fail more often than humans.

RQ1 – What Tasks Do Agents vs Humans Work On?



- Agents handle generative or patch-style work:
  - New feature
  - Documentation
  - Tests
- Humans handle infrastructure & maintenance
  - Requires more general context

RQ1 – Where Do Agents Fail vs Humans?

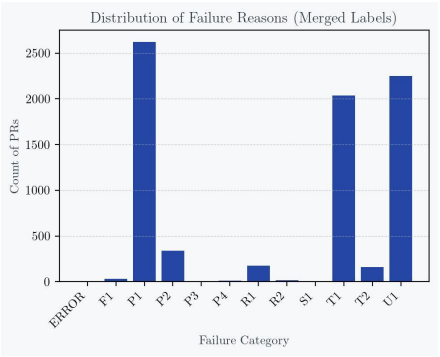


- Agents
  - Strong on structured, repetitive tasks, e.g., docs and tests
  - Struggle on context-heavy maintenance work, especially code style
- Humans
  - Very strong on style and atypical PRs ('other' category)
  - Struggle with test-writing and docs

RQ2: Evaluating model agreement

- Models agree for only ~45% labels.
- Upon analyzing:
  - Qwen2.5 tends to classify ambiguous failures as P1 (low priority).
  - Structured disagreement → systemic bias.
- Fix:
  - When both models label a failure as P1 → true label is P1.
  - If one model labels failure as P1 → pick other model's label.

RQ2 – Classifying Why Agent PRs Fail



- Most Agent PRs fail due to low priority.
- Technical failures result from CI failure.
- Social conflicts and spam are negligible.

RQ3 – How Task Type, Size, and Repo Context Affect Success

- Task type vs failure reason:
  - Low-impact tasks tend to fail silently.
  - Code-changing tasks fail mainly at the CI gate.
- PR Size and Architectural Footprint:
  - Agents perform better when PRs are narrow in scope and easy to review.
  - Large, single-commit diffs across many files more likely to fail.
- Repository Characteristics:
  - Agents struggle in large, high-visibility projects with strict review and lots of competing PRs.

WHAT NOW?

Future Work

- Using better LLMs for classification
- Repeating study across richer datasets
- Creating manually-labelled dataset for failure reasons
- Improving taxonomy to minimize unknown labels

Takeaways and Recommendations

- Agents perform best when:
  - Project scope and priority is clearly defined
  - Repetitive tasks such as boilerplate code, documentation, writing tests
  - PRs are small, narrowly scoped, and include fewer changes
- Humans perform best when:
  - Working on atypical problems
  - Tasks requiring lots of context (performance enhancements, CI configuration)
- Most PR failures result from low priority or CI failure