SWE-Bench Agent Performance Report

Accuracy Results

The agent successfully resolved 10 out of 20 bug instances, achieving a 50% success rate. All 20 instances produced valid patches with no errors or empty submissions, demonstrating that the agent consistently attempts to fix issues even when the solutions don't fully pass all tests.

The agent showed strong performance on Django framework issues (5 out of 11 resolved) and symbolic mathematics libraries (2 out of 2 SymPy issues resolved). It struggled more with documentation generation tools (0 out of 3 Sphinx issues resolved) and some complex architectural problems.

The agent did well on: (1) Straightforward bugs (2) Django framework: Decent performance at 5/11 resolved, likely because Django has consistent patterns and good error messages (3) Sympy: 2/2, where bugs tend to be logic-based rather than architectural

The agent struggled with: (1) Multi-step features: Issues requiring a complete lifecycle (validate → detect → migrate) rather than a single fix (2) Documentation tools: 0/3 success on Sphinx issues, possibly due to less familiarity with documentation generation patterns (3) Situations where the main fix works but breaks other scenarios

Custom Tools

I created five custom tools to help the agent interact with the codebases.

1. `replace_in_file`: Replaces exact text matches in files using Python string replacement instead of shell commands. Takes the old text (copied from the file) and new text, then does a single replacement to avoid accidentally changing multiple locations. Because using shell commands like sed or awk for text replacement can be difficult in certain edge cases (special characters, newlines, etc.), this is a much cleaner way to do it.
2. `show_file`: Displays file contents with line numbers, optionally showing specific line ranges. Line numbers let the agent reference exact locations like "line 522" when identifying what to change.
3. `search_in_files`: Uses grep to recursively search for patterns in the codebase, returning matches with file paths and line numbers (limited to 50 results). Essential for finding code mentioned in bug reports.
4. `find_file`: Locates files by name when the full path is unknown, faster than manually exploring directories.
5. `list_directory`: Shows directory contents to help the agent understand project organization and discover related files.

I made these tools perform simple tasks, as the agent structure and prompt itself is very verbose and complicated. Keeping the actual tools to a minimum helps minimize confusion.

Other Thoughts

I found that I made relatively little progress improving the agent's success rate after the size of my prompt grew beyond a certain length. I also found that the requirement to use gpt-5-mini and therefore maintain a temperature of 1 made the variance of accuracy high as well.

To exceed 50% accuracy repeatedly, I would use: (1) Other models (gpt-4, claude-3.5-sonnet) with temperature=0 support (2) Additional tools for class inspection and call stack tracing (3) Fine-tuning on successful bug fixes