SWE-Bench Can Language Models Resolve Real-World GitHub Issues?

Yifeng He

May 2, 2024

Section 1

Yet another code generation benchmark?

Subsection 1

What wrong with the popular benchmarks?

HumanEval (OpenAI, 2021)

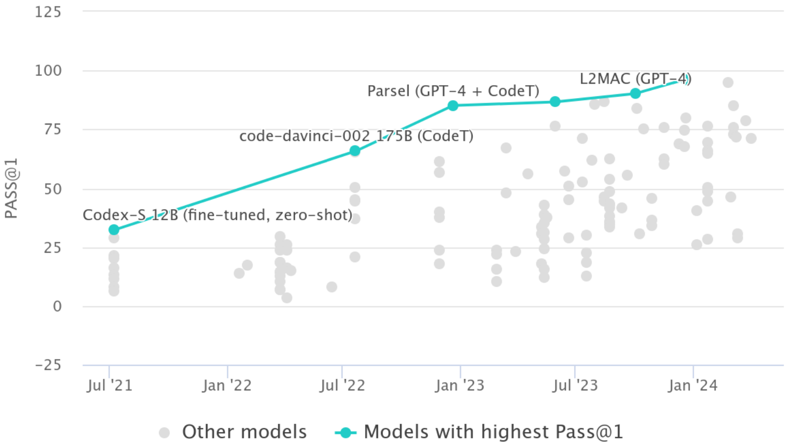


Figure 1: HumanEval Benchmark

```python
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [i + 1 for i in l]
```

Figure 2: HumanEval Problem

Mostly Basic Programming Problems MBPP (Google, 2021)

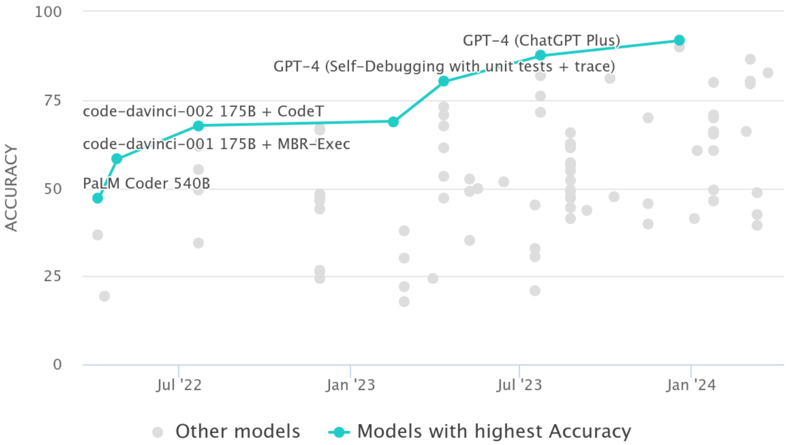


Figure 3: MBPP Benchmark

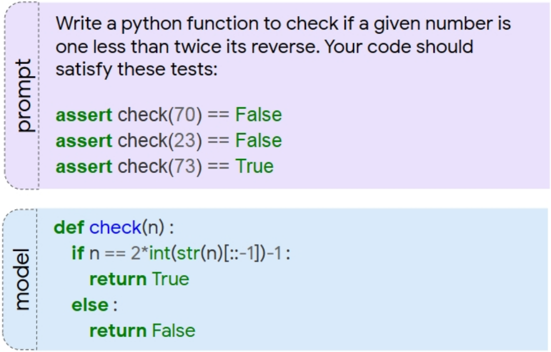


Figure 4: MBPP Problem

SWE-Bench

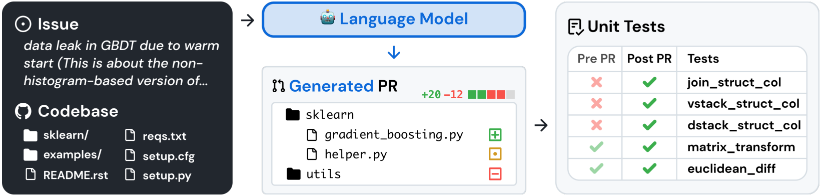


Figure 5: SWE-Bench

- result easily verifiable: pre-defined test suite
- realistic: user-submitted issues and solutions
- diverse: 12 different OSS
- extensible: continuously update the benchmark with new issue/PR pairs

Section 2

Design

Subsection 1

Benchmark construction

Benchmark construction

Figure 6: SWE-Bench construction

1. Data scraping: from 12 popular Python repos, collected $\sim 90,000$ PRs
   - popular repos are better maintained, have clear contributor guidelines, and have better test coverage
2. Attribute-based filtering
   1. *merged* PR with associate issue (makes a task instance)
   2. make changes to a *test* file, indicate that this PR contribute solve the issue
3. Execution-based filtering: at least one *fail-to-pass* test

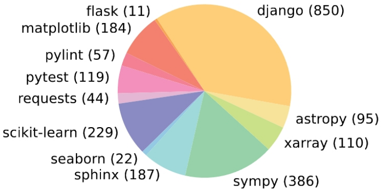


Figure 3: Distribution of SWE-bench tasks (in parenthesis) across 12 open source GitHub repositories that each contains the source code for a popular, widely downloaded PyPI package.

Table 1: Average and maximum numbers characterizing different attributes of a SWE-bench task instance. Statistics are micro-averages calculated without grouping by repository.

		Mean	Max
Issue Text	Length (Words)	195.1	4477
Codebase	# Files (non-test)	3,010	5,890
	# Lines (non-test)	438K	886K
Gold Patch	# Lines edited	32.8	5888
	# Files edited	1.7	31
	# Func. edited	3	36
Tests	# Fail to Pass	9.1	1633
	# Total	120.8	9459

Figure 7: Benchmark Statistics

- Total: 2,294 tasks
- Issue descriptions are short compared to codebase
- Codebase are large with thousands of files
- Pull requests often make changes to multiple files *at once*

Subsection 2

Task Description

Task formulation

- Model input: issue text description + complete codebase
- Task: make an edit to the codebase to resolve the issue
  - edit: generate *patch* files
- Evaluation metrics: apply patch files and see if they resolve the issue
  - resolve the issue: same *fail-to-pass* test

Features

- Real-world SE tasks
- Continually updatable
- Diverse long input
    - issue description is long and in detail, codebase has thousands of files
- Robust evaluation
    - at least 1 *fail-to-pass* test
    - 40% tasks have more than 2
    - a median of 51 additional tests to make check other functionality
- Cross-context code editing
- Wide scope for possible solutions

Challenges

1. A codebase has thousands of files, how can it be fit in the context window?

Section 3

Experiments

Subsection 1

Results

Retrieval for context

1. Sparse retrieval: BM25 (bag-of-words) to retrieve files relevant to issue description
2. "Oracle" retrieval: files edited by the PR

Table 3: BM25 recall with respect to oracle files for different maximum context lengths.

	BM25 Recall		
	13k	27k	50k
Avg.	29.58	44.41	51.06
All	26.09	39.83	45.90
Any	34.77	51.27	58.38

Figure 8: BM25 recall

In $27k$ token limits, superset for $\sim 40\%$ instances, mutually exclusive set for about half of the instances.

Benchmark Results

Table 5: We compare models against each other using the BM25 retriever as described in Section 4. *Due to budget constraints we evaluate GPT-4 on a 25% random subset of SWE-bench.

Model	% Resolved	% Apply
Claude 2	**1.96**	43.07
ChatGPT-3.5	0.17	26.33
GPT-4*	0.00	14.83
SWE-Llama 7b	0.70	51.74
SWE-Llama 13b	0.70	**53.62**

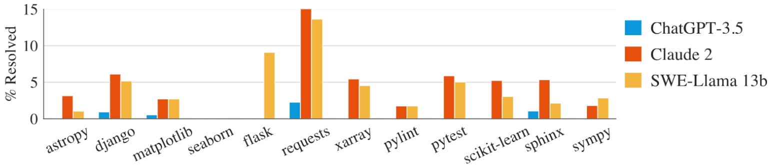


Figure 4: Resolution rate for three models across the 12 repositories represented in SWE-bench in the "Oracle" retrieval setting.

Model	% Resolved	Date	Logs	Trajs	Verified?
SWE-agent + GPT 4	12.47	2024-4-2	🔗	🔗	✓
RAG + Claude 3 Opus	3.79	2024-4-2	🔗	-	✓
RAG + Claude 2	1.96	2023-10-10	🔗	-	✓
RAG + GPT 4	1.31	2024-4-2	🔗	-	✓
RAG + SWE-Llama 13B	0.70	2023-10-10	🔗	-	✓
RAG + SWE-Llama 7B	0.70	2023-10-10	🔗	-	✓
RAG + ChatGPT 3.5	0.17	2023-10-10	🔗	-	✓

Figure 9: Newest results on SWE-Bench

Subsection 2

What factors impact the difficulty of SWE tasks?

Can LLM find the bug in long context?

Figure 5: We compare the performance of Claude 2 on tasks partitioned by total input length and by only the issue length.

Table 6: We show the results for the "Oracle"-collapsed retrieval setting, which uses oracle files but collapses code that isn't directly modified by the PR ± 15 lines.

Model	"Oracle"-collapsed	
	Resolved	Applied
ChatGPT-3.5	1.09	40.93
Claude 2	**5.93**	**68.18**
GPT-4	3.40	48.65

Figure 10: Difficulty correlates with context length

Have the LLM seen this code already?

Table 7: We compare performance on task instances from before and after 2023 in the "Oracle" retrieval setting. Most models show little difference in performance. *Due to budget constraints, GPT-4 is evaluated on a 25% random subset of SWE-bench tasks, which may impact performance.

	Claude 2	ChatGPT-3.5	GPT-4*	SWE-Llama 7b	SWE-Llama 13b
Before 2023	**4.87**	0.49	**1.96**	2.95	**3.98**
After 2023	4.23	**0.77**	0.0	**3.46**	3.85

Figure 11: Difficulty does not correlate with issue resolution date

Does poor root cause localization increase the difficulty?

- Fine-tuned SWE-Llama based on CodeLlama using "oracle" retrieval
- SWE-Llama is sensitive to context distribution shifts
  - perform surprisingly poorly with BM25 retrieved context

Is generating patch file hard?

- the models are trained on whole files instead of patch files, so hard to generate well-formatted patch files?
- Let the models generate wholes, but results are even worse
  - Claude2 scores 4.8% $\rightarrow$ 2.2% with "oracle" retrieval

Is LLMs making the solution over complex?

Table 8: Average edits of model generated patches in the "oracle" retrieval setting across successfully applied patches. For the task instances specific to each model, we calculate the same statistics across the gold patches. Avg Gold shows statistics macro-averaged over each models' respective gold patches. All Gold shows statistics for all gold patches unconditioned on model performance.

Model	Total Lines	Added	Removed	Functions	Files
Claude 2	19.6	4.2	1.9	1.1	1.0
Gold	44.1	12.0	5.8	2.1	1.2
ChatGPT-3.5	30.1	3.8	2.7	1.6	1.0
Gold	39.6	9.5	6.1	1.9	1.2
GPT-4	20.9	4.4	1.5	1.0	1.0
Gold	33.6	8.4	3.8	1.9	1.1
SWE-Llama 13b	17.6	1.6	1.2	1.2	1.1
Gold	37.8	10.0	4.4	1.9	1.1
SWE-Llama 7b	16.7	1.3	1.2	1.2	1.1
Gold	40.2	11.3	4.9	1.9	1.1
Avg Gold	39.1	10.2	5.0	1.9	1.1
All Gold	74.5	22.3	10.5	3.0	1.7

Figure 12: Language models tend to generate shorter, simpler edits

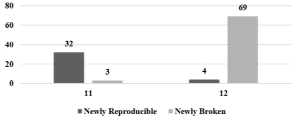
Section 4

Discussion

Is this benchmark really maintainable?

Breakage Frequency

- 1,124 out of 1,795 artifacts broke at least once
- 275 artifacts broken multiple times
- On average, we have 38 newly reproducible and 32 newly broken artifacts in each test suites



Lessons Learned

- All software defect datasets suffer from software breakages, especially for automatically constructed ones
- Most of software breakages are involved with issues related to software dependencies
- Dependency caching and artifact isolation effectively prevent software breakages and ensure long-term reproducibility

On the Reproducibility of Software Defect Datasets (ICSE 2023)