

---

# 03

## Claude 3.5 Sonnet で SWE-bench Verified のベースラインを 引き上げる

— *Raising the bar on SWE-bench Verified with Claude 3.5 Sonnet* —

公開日	2025-01-06
原題	Raising the bar on SWE-bench Verified with Claude 3.5 Sonnet
著者	Anthropic Engineering Team
原文	<a href="https://www.anthropic.com/engineering/swe-bench-sonnet">https://www.anthropic.com/engineering/swe-bench-sonnet</a>
翻訳	Claude(機械翻訳/Anthropic)
編集	2026-04-09

# Claude 3.5 Sonnet で SWE-bench Verified のベースラインを引き上げる

---

私たちの最新モデルであるアップグレード版の [Claude 3.5 Sonnet](#) は、ソフトウェアエンジニアリング評価 [SWE-bench Verified](#) で 49% を達成し、従来の最先端モデルの 45% を上回りました。本稿ではこのモデルの周囲に作った「エージェント」を解説し、開発者が [Claude 3.5 Sonnet](#) から最良のパフォーマンスを引き出せるよう手助けすることを目指します。

[SWE-bench](#) は、モデルが実世界のソフトウェアエンジニアリングタスクを完了できる能力を評価する AI ベンチマークです。具体的には、人気のオープンソース Python リポジトリの GitHub Issue をモデルがどう解決するかをテストします。ベンチマーク内の各タスクでは、Issue が解決される直前の状態で Python 環境とリポジトリのチェックアウト(ローカル作業コピー)がモデルに与えられます。モデルはその上でコードを理解し、修正し、テストし、提案する解を提出する必要があります。

各解は、元の GitHub Issue を閉じたプルリクエストに含まれる本物のユニットテストで評価されます。これは、AI モデルが PR の元の人間の作者と同等の機能を実現できたかどうかを見るものです。

SWE-bench が評価するのはモデル単体ではなく、「エージェント」システム全体です。ここでいう「エージェント」とは、AI モデルと、その周囲のソフトウェア足場(scaffolding)の組み合わせを指します。この足場は、モデルへのプロンプト生成、モデル出力のパース、モデルの前のアクション結果を次のプロンプトに組み込む対話ループの管理を担います。SWE-bench におけるエージェントの性能は、同じモデルを使っても足場次第で大きく変わり得ます。

大規模言語モデルのコーディング能力を測るベンチマークは他にも多数ありますが、SWE-bench が人気を集めているのには理由があります。

1. 競技やインタビュー風の問題ではなく、実際のプロジェクトの本物のエンジニアリングタスクを使っていること。
2. まだ飽和していない。改善余地がたっぷりある。執筆時点で [SWE-bench Verified](#) の完走率 50% を超えたモデルはない(アップデート版 [Claude 3.5 Sonnet](#) は 49%)。
3. モデル単体ではなく「エージェント」全体を測っている。オープンソース開発者やスタートアップは、同じモデルの周りの足場を最適化することで性能を大きく改善してきた実績がある。

なお、元の [SWE-bench](#) データセットには GitHub Issue 外の追加文脈なしには解けないタスク(例: 返すべき特定のエラーメッセージに関するもの)が含まれています。[SWE-bench Verified](#) は [SWE-bench](#) の 500 問のサブセットで、人間によって解ける問題のみが選別されています。コーディングエージェントの性能を最も明

確に測れるため、本稿ではこのベンチマークを指します。

## 最先端の達成

### ツール使用エージェント

アップデート版 Claude 3.5 Sonnet 向けにエージェント足場を作るうえでの私たちの設計哲学は、できる限り多くの制御を言語モデル自身に委ね、足場を最小限に保つというものでした。エージェントは、プロンプト、bash コマンドを実行する **Bash Tool**、ファイルとディレクトリを閲覧・編集する **Edit Tool** を持ちます。私たちはモデルが自分で「完了」と宣言するか、20 万トークンのコンテキスト長を超えるまでサンプリングを続けます。この足場は、モデルが問題への取り組み方を特定のパターンやワークフローにハードコードされず、自分の判断で決められるようにしています。

プロンプトはモデルに推奨される手順を示しますが、このタスクには過度に長くも詳細でもありません。モデルはステップ間の遷移を自由に選べ、厳密で離散的な遷移を強いられることはありません。トークン消費を気にしなくてよいなら、モデルに長い応答を書くよう明示的に促すと性能が上がることもあります。

以下のコードは、私たちのエージェント足場のプロンプトです。

```
<uploaded_files>
{location}
</uploaded_files>
```

I've uploaded a python code repository in the directory {location} (not in /tmp/inputs). Consider the following PR description:

```
<pr_description>
{pr_description}
</pr_description>
```

Can you help me implement the necessary changes to the repository so that the requirements specified in the <pr\_description> are met?

I've already taken care of all changes to any of the test files described in the <pr\_description>. This means you DON'T have to modify the testing logic or any of the tests in any way!

Your task is to make the minimal changes to non-tests files in the {location} directory to ensure the <pr\_description> is satisfied.

Follow these steps to resolve the issue:

1. As a first step, it might be a good idea to explore the repo to familiarize yourself with its structure.
2. Create a script to reproduce the error and execute it with `python <filename.py>` using the BashTool, to confirm the error
3. Edit the sourcecode of the repo to resolve the issue
4. Rerun your reproduce script and confirm that the error is fixed!
5. Think about edgecases and make sure your fix handles them as well

Your thinking should be thorough and so it's fine if it's very long.

モデルの最初のツールは bash コマンドの実行です。スキーマは単純で、環境で実行するコマンドを1つだけ受け取ります。ただし、ツールの説明文にはより重きを置いています。入力のエスケープの仕方、インターネットが使えないこと、バックグラウンドでのコマンド実行方法など、モデル向けの詳細な指示を含めていません。

次に Bash Tool の仕様を示します。

```

{
  "name": "bash",
  "description": "Run commands in a bash shell\n
* When invoking this tool, the contents of the \"command\" parameter does NOT need t
o be XML-escaped.\n
* You don't have access to the internet via this tool.\n
* You do have access to a mirror of common linux and python packages via apt and pi
p.\n
* State is persistent across command calls and discussions with the user.\n
* To inspect a particular line range of a file, e.g. lines 10-25, try 'sed -n 10,25p
/path/to/the/file'.\n
* Please avoid commands that may produce a very large amount of output.\n
* Please run long lived commands in the background, e.g. 'sleep 10 &' or start a ser
ver in the background.",
  "input_schema": {
    "type": "object",
    "properties": {
      "command": {
        "type": "string",
        "description": "The bash command to run."
      }
    },
    "required": ["command"]
  }
}

```

モデルの2つめのツール(Edit Tool)はこれよりずっと複雑で、ファイルの閲覧・作成・編集に必要なものがすべて揃っています。こちらにもツールの説明文には、使い方に関する詳細情報を含めています。

私たちはこれらのツール説明と仕様の作成に、多種多様なエージェントタスクを通じて多くの労力を注ぎました。モデルが仕様を誤解しそうな点や、ツール使用上の落とし穴を洗い出すためにテストし、それらを先回りして潰せるよう説明文を編集しました。人間向けのツールインターフェースに大きな労力がかけられるのと同様に、モデル向けのツールインターフェースにももっと多くの注意が払われるべきだと考えています。

以下のコードは Edit Tool の説明文です。

```

{
  "name": "str_replace_editor",
  "description": "Custom editing tool for viewing, creating and editing files\n
* State is persistent across command calls and discussions with the user\n
* If `path` is a file, `view` displays the result of applying `cat -n`. If `path` is
a directory, `view` lists non-hidden files and directories up to 2 levels deep\n
* The `create` command cannot be used if the specified `path` already exists as a fi
le\n
* If a `command` generates a long output, it will be truncated and marked with `` \n
* The `undo_edit` command will revert the last edit made to the file at `path`\n
\n
Notes for using the `str_replace` command:\n
* The `old_str` parameter should match EXACTLY one or more consecutive lines from th
e original file. Be mindful of whitespaces!\n
* If the `old_str` parameter is not unique in the file, the replacement will not be
performed. Make sure to include enough context in `old_str` to make it unique\n
* The `new_str` parameter should contain the edited lines that should replace the `o
ld_str`,
...

```

性能改善の一手段は、ツールを「エラーに強くする」ことでした。たとえば、エージェントがルートディレクトリから移動した後に相対パスで失敗することがありました。これを防ぐため、ツールは常に絶対パスを要求するようになりました。

既存ファイルへの編集を指定する方式をいくつか試したなかで、最も信頼できたのは文字列置換でした。モデルが対象ファイル内の `old_str` を `new_str` で置き換える指示を出し、`old_str` が 1 つだけ一致するときだけ置換が実行されます。一致が多すぎたり少なすぎたりすれば、モデルに適切なエラーメッセージを返して再試行させます。

Edit Tool の仕様は以下のとおりです。

```

...
  "input_schema": {
    "type": "object",
    "properties": {
      "command": {
        "type": "string",
        "enum": ["view", "create", "str_replace", "insert", "undo_edit"],
        "description": "The commands to run. Allowed options are: `view`, `create`, `str_replace`, `insert`, `undo_edit`."
      },
      "file_text": {
        "description": "Required parameter of `create` command, with the content of the file to be created.",
        "type": "string"
      },
      "insert_line": {
        "description": "Required parameter of `insert` command. The `new_str` will be inserted AFTER the line `insert_line` of `path`.",
        "type": "integer"
      },
      "new_str": {
        "description": "Required parameter of `str_replace` command containing the new string. Required parameter of `insert` command containing the string to insert.",
        "type": "string"
      },
      "old_str": {
        "description": "Required parameter of `str_replace` command containing the string in `path` to replace.",
        "type": "string"
      },
      "path": {
        "description": "Absolute path to file or directory, e.g. `/repo/file.py` or `/repo`.",
        "type": "string"
      },
      "view_range": {
        "description": "Optional parameter of `view` command when `path` points to a file. If none is given, the full file is shown. If provided, the file will be shown in the indicated line number range, e.g. [11, 12] will show lines 11 and 12. Indexing at 1 to start. Setting `[start_line, -1]` shows all lines from `start_line` to the end of the file.",
        "items": {
          "type": "integer"
        },
        "type": "array"
      }
    },
    "required": ["command", "path"]
  }

```

```
}  
}
```

## 結果

総じて、アップグレード版 Claude 3.5 Sonnet は、旧モデルや[従来の最先端モデル](#)と比べて、推論・コーディング・数学能力が向上しています。エージェント能力も改善しており、今回のツールと足場はその改善を最大限に活かすのに役立っています。

モデル	Claude 3.5 Sonnet (new)	従来 SOTA	Claude 3.5 Sonnet (old)	Claude 3 Opus
SWE-bench Verified スコア	49%	45%	33%	22%

*SWE-bench Verified* における各モデルのスコア(本稿のエージェント足場を使用)

## エージェントの振る舞いの例

ベンチマーク実行にあたっては、エージェントコードの基盤として [SWE-Agent](#) フレームワークを使用しました。以下のログでは、エージェントのテキスト出力・ツール呼び出し・ツール応答を **THOUGHT・ACTION・OBSERVATION** として描画しています(モデルを固定の順番に強制しているわけではありません)。

以下のコードブロックは、Sonnet 3.5 が SWE-bench の問題を解く典型的な流れです。

最初のブロックは、モデルに与えられた初期プロンプトの一部です。`{pr_description}` には SWE-bench タスクの実データが入ります。重要な点として、このタスクには問題の再現手順が含まれており、モデルが調査を始める貴重な出発点になります。

```

<uploaded_files>
/repo
</uploaded_files>
I've uploaded a python code repository in the directory /repo (not in /tmp/inputs).
Consider the following PR description:

<pr_description>
linear_model.RidgeClassifierCV's Parameter store_cv_values issue
#### Description
Parameter store_cv_values error on sklearn.linear_model.RidgeClassifierCV

#### Steps/Code to Reproduce
import numpy as np
from sklearn import linear_model as lm

#test database
n = 100
x = np.random.randn(n, 30)
y = np.random.normal(size = n)

rr = lm.RidgeClassifierCV(alphas = np.arange(0.1, 1000, 0.1), normalize = True,
                          store_cv_values = True).fit(x, y)

#### Expected Results
Expected to get the usual ridge regression model output, keeping the cross validation predictions as attribute.

#### Actual Results
TypeError: __init__() got an unexpected keyword argument 'store_cv_values'

lm.RidgeClassifierCV actually has no parameter store_cv_values, even though some attributes depends on it.

```

モデルは応答し、まず Edit Tool でリポジトリ構造を閲覧します。THOUGHT と ACTION にモデルのテキスト出力とツール呼び出しの引数が、OBSERVATION にツール出力の一部が見られます。

THOUGHT:

I'll help you implement the changes to satisfy the PR description. Let's follow the steps you outlined.

1. First, let's explore the repo structure:

ACTION (str\_replace\_editor):

command:

view

path:

/repo

OBSERVATION:

Here's the files and directories up to 2 levels deep in /repo, excluding hidden items:

```
/repo
/repo/PULL_REQUEST_TEMPLATE.md
/repo/ISSUE_TEMPLATE.md
/repo/sklearn
/repo/sklearn/decomposition
/repo/sklearn/kernel_ridge.py
/repo/sklearn/model_selection
/repo/sklearn/covariance
/repo/sklearn/cluster
/repo/sklearn/base.py
/repo/sklearn/dummy.py
/repo/sklearn/naive_bayes.py
...
```

リポジトリ構造を把握したモデルは、問題を再現して修正を確認するための新しいスクリプトを Edit Tool で作成します。

THOUGHT:

Based on the repository structure, we'll need to modify the Ridge Classifier in the `sklearn/linear\_model` directory.

Let's create a script to reproduce the error:

ACTION (str\_replace\_editor):

command:

create

path:

/repo/reproduce\_error.py

file\_text:

```
import numpy as np
```

```
from sklearn import linear_model as lm
```

```
#test database
```

```
n = 100
```

```
x = np.random.randn(n, 30)
```

```
y = np.random.normal(size = n)
```

```
rr = lm.RidgeClassifierCV(alphas=np.arange(0.1, 1000, 0.1), normalize=True,  
                          store_cv_values=True).fit(x, y)
```

OBSERVATION:

File created successfully at: /repo/reproduce\_error.py

モデルは次に Bash Tool でそのスクリプトを実行し、タスクの問題をきちんと再現します。

THOUGHT:

Let's verify that we can reproduce the error:

ACTION (bash):

command:

```
python3 /repo/reproduce_error.py
```

OBSERVATION:

```
Traceback (most recent call last):
```

```
  File "/repo/reproduce_error.py", line 10, in <module>
```

```
    store_cv_values=True).fit(x, y)
```

```
TypeError: __init__() got an unexpected keyword argument 'store_cv_values'
```

ここから先、モデルは Edit Tool でリポジトリのソースコードを変更し、スクリプトを再実行して問題が解決したかを確認します。

#### THOUGHT:

The error is reproducible. Looking at the code, I can see that the issue is that while the `RidgeClassifierCV` class inherits from `\_BaseRidgeCV`, it fails to pass the `store\_cv\_values` parameter from its constructor through to the base class constructor.

Let's fix this:

ACTION (str\_replace\_editor):

command:

str\_replace

path:

/repo/sklearn/linear\_model/ridge.py

old\_str:

```
def __init__(self, alphas=(0.1, 1.0, 10.0), fit_intercept=True,
              normalize=False, scoring=None, cv=None, class_weight=None):
    super(RidgeClassifierCV, self).__init__(
        alphas=alphas, fit_intercept=fit_intercept, normalize=normalize,
        scoring=scoring, cv=cv)
    self.class_weight = class_weight
```

new\_str:

```
def __init__(self, alphas=(0.1, 1.0, 10.0), fit_intercept=True,
              normalize=False, scoring=None, cv=None, class_weight=None,
              store_cv_values=False):
    super(RidgeClassifierCV, self).__init__(
        alphas=alphas, fit_intercept=fit_intercept, normalize=normalize,
        scoring=scoring, cv=cv, store_cv_values=store_cv_values)
    self.class_weight = class_weight
```

この例では、モデルは 12 ステップ動いた後、提出準備が整ったと判断しました。タスクのテストはその後正常に走り、モデルの解が問題を解決したことが検証されました。中にはモデルが 100 ターン以上動いてから提出するタスクもあれば、コンテキストを使い切るまで試行を続けるケースもあります。

アップデート版 Claude 3.5 Sonnet の試行を旧モデルと比較してレビューすると、3.5 Sonnet はより頻繁に自己修正を行います。また、同じ間違いを繰り返し続けるのではなく、複数の異なる解を試す能力も示しています。

## 課題

SWE-bench Verified は強力な評価ですが、単純な 1 ターンの eval より実行が複雑でもあります。私たちがこの eval を使って直面した課題のうち、他の AI 開発者にも起こりそうなものを紹介します。

1. **所要時間と高いトークンコスト**。前述の例は 12 ステップで完了した成功ケースですが、多くの成功実行はモデルが問題を解くのに何百ターンもかかり、10 万トークンを超えました。アップデート版 Claude 3.5 Sonnet は粘り強く、時間さえあれば多くの問題で解決経路を見つけ出しますが、それは高くつくこともあります。
2. **採点**。失敗タスクを調査する中で、モデルは正しく振る舞っていたのに環境構築の問題や、インストールパッチが二重適用されてしまう問題に阻まれていたケースが見つかりました。AI エージェントの性能を正確に把握するには、こうしたシステム側の問題を解決することが重要です。
3. **隠しテスト**。モデルは自分が採点される対象のテストを見られないため、実際には失敗しているタスクを「成功した」と「思い込む」ことがしばしば起きます。そうした失敗の一部は、モデルが問題を誤った抽象レベルで解いてしまったことによるもの(深いリファクタの代わりに絆創膏を貼った、など)です。もう少し不公平に感じられる失敗もあります。問題自体は解いているものの、元タスクのユニットテストに合致しないケースです。
4. **マルチモーダル**。アップデート版 Claude 3.5 Sonnet は優れたビジョン・マルチモーダル能力を持ちますが、私たちは今回、ファイルシステム上のファイルや URL 参照の画像をモデルが閲覧できる手段を実装しませんでした。そのため、ある種のタスク(特に Matplotlib 関連)のデバッグは非常に難しく、モデルの幻覚も起きやすくなりました。この領域には開発者が改善できる低くて熟した果実があります。SWE-bench 自身も新しい[マルチモーダルタスクに特化した eval](#) を出しました。近い将来、Claude を使った開発者がこの eval でより高いスコアを達成するのを楽しみにしています。

アップグレード版 Claude 3.5 Sonnet は、シンプルなプロンプトと 2 つの汎用ツールだけで SWE-bench Verified で 49% を達成し、従来の最先端(45%)を上回りました。新しい Claude 3.5 Sonnet で構築する開発者たちが、本稿で最初に示した水準を越えて、SWE-bench スコアを改善する新しい良い方法をすぐに見つけ出すと私たちは確信しています。

## 謝辞

Erik Schluntz が SWE-bench エージェントを最適化し、本稿を執筆しました。Simon Biggs、Dawn Drain、Eric Christiansen はベンチマーク実装を手伝ってくれました。Shauna Kravec、Dawn Drain、Felipe Rosso、Nova DasSarma、Ven Chandrasekaran そのほか多くの人々が、Claude 3.5 Sonnet をエージェント型コーディングに優れさせる訓練に貢献しました。