

Terminal-Bench

端末での作業遂行を評価するデータセット
公開版 2.0 の課題・採点・モデル評価

公式リポジトリ

Terminal-Bench 2.0

🔗



Docs

Terminal-Bench is a popular benchmark for measuring the capabilities of agents and language models to perform valuable work in containerized environments. Tasks include assembling proteins for synthesis, debugging async code, and resolving security vulnerabilities.

It is used by virtually all frontier labs.

Getting Started

🔗

First, install [Harbor](#), our package for evaluating and optimizing agents:

```
uv tool install harbor
```

or

```
pip install harbor
```

You should now be able to run TB 2.0! Test it by running the dataset oracle solutions in docker containers locally:

```
uv run harbor run --dataset terminal-bench2.0 \
  --agent oracle \
  --n-concurrent 4
```

This command automatically downloads the tasks for the benchmark. You can view them at <https://github.com/laude-institute/terminal-bench-2>

You can also run using Terminus:

```
export ANTHROPIC_API_KEY=<YOUR-KEY>
uv run harbor run --dataset terminal-bench2.0 \
  --agent terminus-2 \
  --model anthropic/claude-opus-4-1 \
  --n-concurrent 4
```

1. 端末で行う作業と実例
2. 採点方法とモデルの成績
3. 課題の作成方法と全体像
4. 利用の変遷と実行方法

1. 端末で行う作業と実例

Terminal-Bench は端末での作業完了を測る

- 端末は、文字でコマンドを入力し、ファイル編集やプログラム実行を行う操作環境である。
- Terminal-Bench では、指示と作業環境を受け取った AI エージェントが、端末を使って課題を完了させる。
- 公開版 2.0 は 89 課題。ソフトウェア開発、システム管理、科学計算、データ処理などを扱う。

エージェントは操作と結果の確認を繰り返す

- 言語モデルが、次に行うコマンドや編集内容を決める。エージェントは、その判断を道具の呼出しへつなぐ仕組みである。
- ファイルやエラーを読み、修正し、プログラムを動かし、結果を確認する。この往復を時間制限まで行える。
- 採点は、作業後のファイル・プログラムの動作・設定などをテストして行う。

1 課題に指示・環境・合格条件がそろう

構成	与えられるもの・役割
指示文	実現する動作、出力先、守る条件
実行環境	Linux 環境と、入力ファイル・依存ソフト
採点テスト	完成後の状態を検査するプログラム
参照解法	人が作成した、合格する解法

実例 1 : 中断時にも後始末する非同期処理

指示文の原文（英語・抜粋）

```
Create a Python function called `async run_tasks(tasks: list[Callable[[],  
Awaitable[None]]], max_concurrent: int) -> None`  
Sometimes I cancel runs via keyboard interrupt but I want the tasks'  
cleanup code to still run.
```

要旨： Python で複数の非同期ジョブを並行実行する関数を作る。同時実行数を制限し、キーボードで中断されても各ジョブの後始末を行う。

実例 1 は並行実行と中断処理を検査する

判定項目	公開テストの内容
成果物	/app/run.py から run_tasks を読み込める
並行実行	3 秒のジョブ 2 件を、上限 2 なら並行して完了
上限の順守	上限 1 なら、2 件の実行に 6 秒以上かかる
中断時の処理	SIGINT で中断し、各ジョブの後始末を確認

実例 2：更新ログからデータベースを復元する

指示文の原文（英語・抜粋）

I have a database in WAL (Write-Ahead Logging) mode in /app/.
However, the WAL file appears to be corrupted or encrypted. When
you try to access the database, SQLite may only show the base data (5
records) instead of all 11 records that should be there.
You should recover all 11 records total.

要旨： SQLite の更新ログ WAL を修復し、更新分も含めた 11 件を
JSON へ保存する。 WAL は、変更を先に記録するファイルである。

実例 2 は復元した 11 件の内容を採点する

判定項目	公開テストの内容
保存先	/app/recovered.json
構造と順序	id ・ name ・ value を持つ配列を id 順にする
全件の復元	id が 1 ～ 11 の、重複のない 11 件
更新の反映	id=1 の value は 150 、 id=2 は 250

実例 3 : COBOL の処理を Python へ移す

指示文の原文（英語・抜粋）

Your task is to re-implement the functionality of program.cbl in Python. You must create a new Python script located at /app/program.py, which performs the exact same operations as the COBOL program.

要旨：業務用言語 COBOL のプログラムを読み、同じ入力から同じデータファイルを作る Python 版を /app/program.py に実装する。

実例 3 は入力に対するファイル更新を照合する

対象	役割・合格条件
INPUT.DAT	取引などの入力を与える
ACCOUNTS.DAT	利用者ごとの残高を更新する
BOOKS.DAT	書籍と利用者の対応を更新する
TRANSACTIONS.DAT	取引の記録を追加する

同じ初期状態と入力列から、3 ファイルの内容一致を確認する。

2. 採点方法とモデルの成績

全テストに合格した試行を成功と数える

- 1 回の試行でエージェントが作業し、その終了後に採点テストを実行する。
- 全テストに合格すれば成功の 1、条件を満たさなければ 0 を与える。課題には作業時間の上限も設定されている。
- 複数の課題・試行の成功率を平均し、解決率（resolution rate）を百分率で報告する。

説明用の 3 課題では、2 件成功で 66.7%

説明用の試行	テストの結果	得点
非同期処理の実装	すべて合格	1
データベース復元	復元件数が不足	0
COBOL の移植	すべて合格	1

成功 2 件を全 3 件で割り、百分率にすると約 66.7%。

Merrill ら（2026）による 2.0 の評価

モデル	エージェント	解決率
GPT-5.2	Codex CLI	62.9%
Claude Opus 4.5	Terminus 2	57.8%
Gemini 3 Pro	Terminus 2	56.9%

各モデルとエージェントの組を 5 回以上評価。

2026 年の 2.0 評価でも複数系列が使われる

モデル（Google 公表表）	解決率
Gemini 3.1 Pro / High	68.5%
Claude Opus 4.6 / Max	65.4%
GPT-5.2 / xhigh	54.0%

いずれも Terminus-2 による評価。

同じモデルの成績もエージェントで変わる

Gemini 2.5 Pro のエージェント	2.0 の解決率
Terminus 2	32.6%
Mini-SWE-Agent	26.1%
Gemini CLI	19.6%
OpenHands	15.7%

Merrill ら（2026）による Gemini 2.5 Pro の評価。

2.0 と 4.0 はデータセットのバージョン

- 2.0 は、指示文・実行環境・採点テストを公開している 89 課題の版。本資料の実例と分布は、この公開版に基づく。
- 4.0 は、2026 年の GPT-6 Astra や Claude Opus 5.5 の公表評価に使われる後続版である。
- Terminus 2 はエージェントの名前。2.0 の評価では、このエージェントと各モデルを組み合わせで実行する。

The Terminal-Bench Team (2025). Terminal-Bench 2.0. 公式 GitHub 。2026-10-16 参照。 <https://github.com/laude-institute/terminal-bench-2>

Anthropic (2026-09-22). Claude Opus 5.5. Terminal-Bench 4.0. 2026-10-16 参照。 <https://www.anthropic.com/news/claude-opus-5-5>

4.0 は端末での複雑な作業を評価する

- OpenAI は 4.0 の課題として、ソフトウェア開発、システムの設定、データ分析を挙げている。
- Anthropic は、コマンドライン上で複数の手順を実行する、複雑な実務課題の評価と説明している。
- Anthropic が参照する公開リーダーボードは、Claude Code を使い、各課題を 5 回試行して評価する。

OpenAI (2026). GPT-6 Astra: A new generation of intelligence. Terminal-Bench 4.0. 2026-10-16 参照。 <https://openai.com/index/gpt-6-astra/>

Anthropic (2026-09-22). Claude Opus 5.5. Terminal-Bench 4.0. 2026-10-16 参照。 <https://www.anthropic.com/news/claude-opus-5-5>

最新モデルの公表表には 4.0 も採用される

モデル	4.0 の解決率
Claude Opus 5.5	66.4%
GPT-6 Astra	57.9%
GPT-5.6 Sol	37.3%

Opus 5.5 : xhigh effort。 GPT-6 Astra : high effort。

Terminus 2 は端末だけを道具として使う

- Terminus 2 は、モデルへ端末の入出力を渡し、 Bash コマンドで作業を進めるエージェントである。
- コマンドの実行、実行中のプロセスからの出力取得、キーボード入力を繰り返し、対話的な作業も行う。
- Merrill ら（2026）は Terminus 2 に加え、 Claude Code、Codex CLI、Gemini CLI、OpenHands、Mini-SWE-Agent を評価した。

3. 課題の作成方法と全体像

複数分野の実作業を難しい課題として集めた

- コード修正、計算環境の整備、研究用プログラムの実装など、専門家が端末で行う作業を課題にした。
- 各課題には必要なファイルとソフトウェアを用意し、作業結果を自動テストで判定できるようにする。
- 2.0 では、複雑な依存関係や専門知識を要する課題を集め、複数人で指示とテストを審査した。

コード修正や Web 操作から、端末の多分野作業へ

- SWE-bench（2023 年公開）は、GitHub の課題記述から既存リポジトリを修正する評価データである [1]。
- WebArena（2023 年公開）は、買物・掲示板などの Web アプリをブラウザで操作する評価環境である [2]。
- Terminal-Bench は端末を共通の操作面とし、ソフトウェア開発から科学計算まで対象を広げる。

[1] Jimenez et al. (2024). SWE-bench: Can Language Models Resolve Real-World GitHub Issues? ICLR. <https://arxiv.org/abs/2310.06770>

[2] Zhou et al. (2024). WebArena: A Realistic Web Environment for Building Autonomous Agents. ICLR. <https://arxiv.org/abs/2307.13854>

93 人の貢献者から 229 課題を集めた

段階	Merrill ら（2026）の報告
課題の投稿	93 人が 229 課題を作成
選定	作者の難度評価と、審査者の品質評価
最終集合	2.0 に 89 課題を採用
人手審査	3 人が確認し、1 課題当たり計約 3 時間

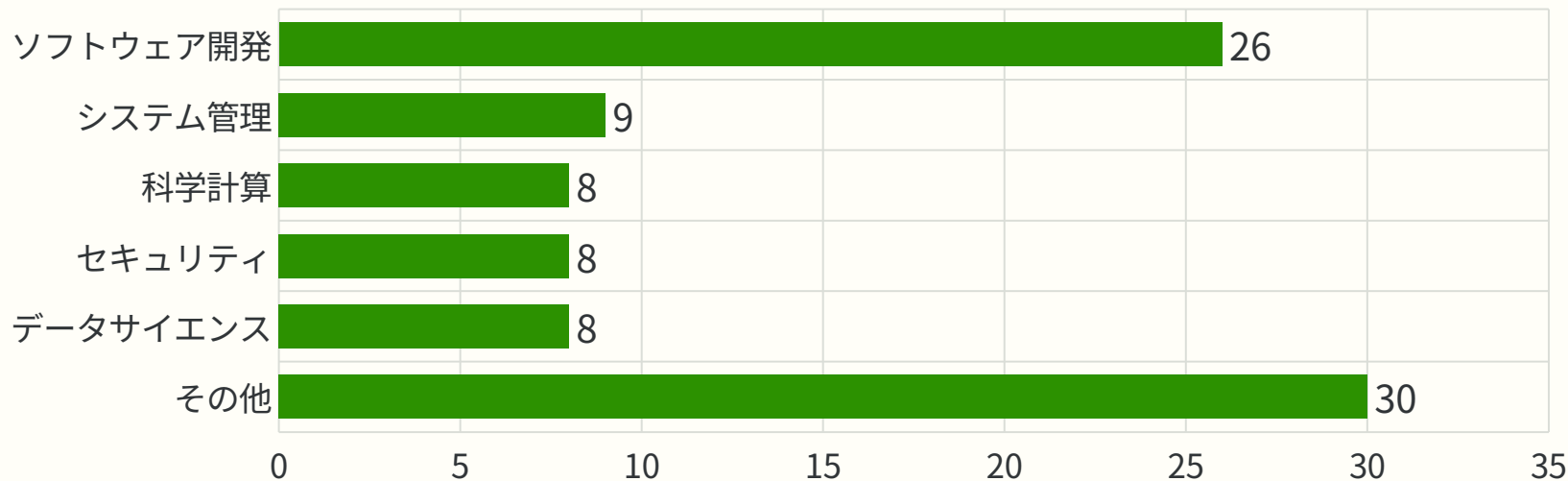
審査では指示・解法・抜け道を確認する

基準	確認する内容
具体性	許容する完成状態とテストの条件が対応する
可解性	人が作った参照解法ですべてのテストに合格する
評価の健全性	未来の Git 履歴などから答えを得る抜け道を除く

自動検査と人手による実行履歴の監査を重ねる

工程	行う確認
投稿時	参照解法は合格し、何もしない場合は失敗
コード・仕様の確認	LLM による検査と、経験ある審査者のレビュー
モデルでの試行	失敗が課題の不備か、解法の失敗かを調べる
採用前の監査	追加の 2 人が実行履歴と抜け道を確認

ソフトウェア開発が 26 課題を占める



公開版 2.0 の全 89 課題。作者が付けた 16 カテゴリを集計。

作者が付けた難度では medium が 55 課題

難度（配布表記）	課題数
easy	4
medium	55
hard	30

作者の所要時間見積もりは専門家で中央値 1 時間

作業者の想定	中央値
その分野の専門家	60 分
分野に詳しくない若手エンジニア	240 分

作業時間の見積もりがある 88 課題の配布メタデータを集計。

51 課題のエージェント実行上限は 15 分以下

実行上限時間	課題数
15 分以下	51
15 分超～ 30 分以下	22
30 分超～ 60 分以下	14
60 分超	2

task.toml の agent.timeout_sec を全 89 課題で集計。

失敗した実行履歴を 3 種類に分類した

分類	具体的な失敗
実行	指定への違反、同じ操作の反復、終了条件の見落とし
一貫性	推論と操作の不一致、文脈の喪失、課題からの逸脱
検証	早すぎる終了、未検証・誤った確認、不十分な確認

Merrill ら（2026）は Terminus 2 の失敗履歴を分析し、LLM で分類した。

4. 利用の変遷と実行方法

2025 年の公開から、新しい課題集合へ展開した

時期・資料	利用と更新
2025 年の公開	Terminal-Bench の端末操作課題を公開
Merrill ら（2026 年 1 月）	89 課題と複数エージェントの評価を報告
2026 年の公式実装	Harbor に Opus 5.5 + 2.0 の実行例
2026 年のモデル公表表	GPT-6 Astra ・ Opus 5.5 の 4.0 評価

課題フォルダには実行と採点の材料が入る

ファイル・フォルダ	内容
instruction.md	エージェントに与える作業指示
task.toml	難度・分野・時間制限・環境設定
environment/	Dockerfile と初期ファイル
tests/	完成状態を検査するコード
solution/	人が書いた参照解法

Harbor で公開版 2.0 を取得して実行する

```
uv tool install harbor
```

```
harbor run --dataset terminal-bench@2.0 \  
  --agent oracle \  
  --n-concurrent 4
```

Harbor は評価の実行基盤。ローカルでは Docker で課題ごとの環境を作り、oracle は公開参照解法を実行する。

同じ課題を Claude Code で評価する

```
harbor run --dataset terminal-bench@2.0 \  
  --agent claude-code \  
  --model anthropic/claude-opus-5-5 \  
  --n-concurrent 4
```

Harbor 公式 README の実行例。モデル API の認証を設定すると、課題の取得、環境の準備、作業、採点を実行する。

配布元・ライセンスと実行結果の保存先

- 公開課題は公式 GitHub で配布され、Harbor のレジストリから terminal-bench@2.0 として取得できる。
- 公開タスクリポジトリは Apache License 2.0 。課題に同梱・参照されるデータやソフトウェアの条件も各配布元で示される。
- Harbor は試行ごとの採点結果、エージェントの実行履歴、実行時間などを保存する。

Terminal-Bench の特徴と利用

- 指示と実行環境を与え、端末を使うエージェントが成果物を作るかをテストする。
- 公開版 2.0 は、93 人の投稿から選び、複数人で審査した 89 課題。分野と作業内容の広さが特徴である。
- Merrill ら（2026）による複数エージェントの比較から、現在のモデル評価と Harbor による実行へ利用が続いている。

Appendix

1. 分野の異なる追加例
2. 分布・実装と採点の詳細
3. 開発者と参考文献

1. 分野の異なる追加例

追加例 1 : 同じ結果を返す SQL を高速化する

指示文の原文（英語・抜粋）

I implemented a sql query but it is not optimized. I have saved it in /app/my-sql-query.sql. Please make the query as efficient as possible while ensuring that the same output is produced.

要旨：単語の概念を整理した Open English WordNet の SQLite データベースを使う。結果を保って SQL を高速化し、 /app/sol.sql へ保存する。

追加例 2 : NumPy に合う拡張モジュールを作る

指示文の原文（英語・抜粋）

Can you help me compile extensions, install pyknotid from source to system's global python environment, while fixing any Numpy compatibility issues so it works with my existing Numpy (2.3.0) versions?

要旨：結び目の計算用ライブラリ pyknotid を修正・ビルドする。テストは NumPy 2.3.0 の保持、3 拡張の読み込み、結び目の計算結果を確認する。

追加例 3 : Git への push を Web 公開へつなぐ

指示文の原文（英語・抜粋）

And have this data then be pushed to a webserver running on port 8080 so if I run `curl http://server:8080/hello.html` then I see the output "hello world".

要旨： Git サーバーを設定し、push した `hello.html` がポート 8080 の Web サーバーから取得できるようにする。テストは実際の push と HTTP 取得を行う。

追加例 4：期間別・重要度別にログを数える

指示文の原文（英語・抜粋）

The severity levels to count are exactly: ERROR, WARNING, and INFO.
The current date is 2025-08-12. Use this as the reference date for all calculations.

要旨：指定日、直近 7 日・30 日、月初から指定日、全期間の 5 区分で集計する。3 重要度 × 5 期間の 15 行を summary.csv に出力し、全行の値を照合する。

追加例 5 : 3 形式の利用者データを統合する

指示文の原文（英語・抜粋）

Merge user data from three different sources with different formats and schemas.

4. Handle conflicts using source priority (source_a > source_b > source_c)

要旨：JSON・CSV・Parquetを読み、user_idと列名をそろえる。
優先順位で値を選び、統合Parquetと競合レポートを保存する。

追加例 6：同義語で LaTeX の行はみ出しを直す

指示文の原文（英語・抜粋）

Ensure that the LaTeX document `main.tex` compiles successfully using the installed TeX distribution and `pdflatex` compiler with no "overfull hbox" warnings. In doing so, the only edits you may make are to replace words in `input.tex` with their specified synonyms in `synonyms.txt` (each line specifies a family of allowed synonyms).

要旨：許可された同義語への置換だけで、行のはみ出し警告を消す。テストは PDF 生成、警告の消失、変更範囲を確認する。

追加例 7 : ラマン分光の 2 ピークを推定する

指示文の原文（英語・抜粋）

You are given the output file of a Raman Setup. We used it to measure some graphene sample. Fit the G and 2D Peak of the spectrum and return the x0, gamma, amplitude and offset of the peaks and write them to a file called "/app/results.json".

要旨：グラフェン試料のスペクトルに G ・ 2D のピークを当てはめ、中心位置・幅・振幅・オフセットを JSON へ保存する。採点はパラメータごとの許容範囲を確認する。

追加例 8：制約に合わせてデータを分割する

指示文の原文（英語・抜粋）

- Maximum 30 files or folders in each directory - Maximum 15MB filesize per file
- Should reconstruct the original file structure and content exactly

要旨：データを再分割する `compress.py` と元へ戻す `decompress.py` を作る。テストは容量・項目数の上限と、復元後のファイル集合・ハッシュを確認する。

追加例 9 : 自然数の加法の交換法則を証明する

指示文の原文 (英語・抜粋)

The theorem states that addition is commutative for natural numbers (forall n m : nat, n + m = m + n).

3. Compile the completed proof using coqc

要旨：証明支援系 Coq で、不完全な帰納法の証明を完成させる。採点は証明の完了と、coqc によるコンパイルを確認する。

追加例 10 : 動画から跳躍の開始・着地を検出する

指示文の原文（英語・抜粋）

```
Write a script, named jump_analyzer.py, and place it in  
`/app/jump_analyzer.py`. The script analyzes MP4 videos of hurdle  
jumpers and extracts performance metrics.
```

要旨：ハードル走の映像を処理し、踏切と着地のフレーム番号を TOML へ出力する。公開テストは開発用・別動画で許容範囲を照合し、使用ライブラリも検査する。

2. 分布・実装と採点の詳細

公開 89 課題のカテゴリ別件数（ 1/2 ）

作者のカテゴリ（日本語）	課題数
ソフトウェア開発	26
システム管理	9
科学計算	8
セキュリティ	8
データサイエンス	8
デバッグ	5
ファイル操作	5
モデル学習	4

公開 89 課題のカテゴリ別件数（2/2）

作者のカテゴリ（日本語）	課題数
数学	4
データ処理	4
機械学習	3
ゲーム	1
個人向け支援	1
最適化	1
データ照会	1
動画処理	1

公開メタデータの所要時間見積もり

対象 88 課題	専門家	若手エンジニア
中央値	60 分	240 分
最小	5 分	10 分
最大	2,400 分	19,200 分

author が記入した expert_time_estimate_min / junior_time_estimate_min。

The Terminal-Bench Team (2025). Terminal-Bench 2.0. 公開 89 課題、改訂 69671fb。2026-10-16 集計。
<https://github.com/laude-institute/terminal-bench-2/tree/69671fbaac6d67a7ef0dfec016cc38a64ef7a77c>

実行時間の上限は課題ごとに設定する

設定名	公開版の範囲・実例
agent.timeout_sec	作業の上限。全 89 課題で 600 ~ 12,000 秒
verifier.timeout_sec	採点テスト側の時間制限
environment	CPU ・ メモリ ・ ストレージと環境の構築条件

参照する改訂で課題を固定して取得できる

```
git clone https://github.com/laude-institute/terminal-bench-2.git
cd terminal-bench-2
git checkout 69671fbaac6d67a7ef0dfec016cc38a64ef7a77c

ls db-wal-recovery
```

Harbor レジストリの terminal-bench@2.0 は、89 課題すべてをこの Git 改訂で指定している。

データ復元のテストは件数と値を確認する

```
assert len(data) == 11
ids = [item["id"] for item in data]
assert ids == list(range(1, 12))

id_to_item = {item["id"]: item for item in data}
assert id_to_item[1]["value"] == 150
assert id_to_item[2]["value"] == 250
```

db-wal-recovery/tests/test_outputs.py からの抜粋。id 順の全 11 件と、WAL に含まれる更新値を照合する。

タスクのソースファイルは約 43.77 MiB

配布物	取得した版の容量・用途
Git 管理ファイル	45,894,095 bytes （ 43.77 MiB ）
コンテナイメージ	Dockerfile または指定イメージを使用
作業用の保存領域	task.toml の環境設定で課題ごとに指定

公開 Git 改訂 69671fb の管理ファイルを集計。 1 MiB = 2^{20} bytes 。

ライセンスと参照先

対象	公開元の指定
Terminal-Bench 2.0 の公開リポジトリ	Apache License 2.0
取得と実行の案内	公式 README / Harbor
各タスクの作業条件	instruction.md と task.toml

3. 開発者と参考文献

大学・研究機関と開発者コミュニティが構築した

- 共同第一著者は Mike A. Merrill と Alexander G. Shaw 。論文発表時の所属は、それぞれ Stanford University と Laude Institute である。
- 最終著者は Stanford University の Ludwig Schmidt 。端末でのエージェント評価と、幅広い技術作業の課題設計を共同で研究した。
- Merrill ら（2026）は 2026 年の ICLR で発表。公開プロジェクトには多数の課題作成者・審査者が参加している。

参考文献

Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces

Mike A. Merrill, Alexander G. Shaw et al. / 2026 / ICLR

89 課題の構築・審査、モデルとエージェントの比較、失敗履歴の分析を報告する原著。

<https://arxiv.org/abs/2601.11868>

参考文献

Terminal-Bench 2.0

The Terminal-Bench Team / 2025 / GitHub

指示、初期環境、参照解法、採点テストを含む 89 課題を公開する。

<https://github.com/laude-institute/terminal-bench-2>

Harbor

Harbor contributors / 2026 参照 / GitHub

端末操作エージェントの環境準備、実行、採点、結果保存を扱う公式基盤。

<https://github.com/laude-institute/harbor>

参考文献

SWE-bench: Can Language Models Resolve Real-World GitHub Issues?

Carlos E. Jimenez et al. / 2024 / ICLR

GitHub の課題記述とリポジトリから、実際のコード修正を評価する先行ベンチマーク。

<https://arxiv.org/abs/2310.06770>

参考文献

WebArena: A Realistic Web Environment for Building Autonomous Agents

Shuyan Zhou et al. / 2024 / ICLR

Web アプリでの情報取得や操作を、実行結果から評価する先行研究。

<https://arxiv.org/abs/2307.13854>

参考文献

Gemini 3.1 Pro

Google DeepMind / 2026 / Model information

Terminus-2 を用いた Terminal-Bench 2.0 の評価値を掲載する。

<https://deepmind.google/models/gemini/pro/>

Claude Opus 5.5

Anthropic / 2026-09-22 / Model announcement

Terminal-Bench 4.0 の性能と推論設定、実行費用の比較を公表する。

<https://www.anthropic.com/news/claude-opus-5-5>

参考文献

GPT-6 Astra: A new generation of intelligence

OpenAI / 2026 / Model announcement

Terminal-Bench 4.0 を含む、端末操作とコード開発の評価結果を公表する。

<https://openai.com/index/gpt-6-astra/>