ScaDS.AI
DRESDEN LEIPZIG
CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Benchmarking Large Language Models for Bio-Image Analysis Code Generation

Robert Haase, Christian Tischer, Jean-Karim Hériché, Nico Scherf

https://www.biorxiv.org/content/10.1101/2024.04.19.590278v3

Bundesministerium für Bildung und Forschung
SACHSEN
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

TECHNISCHE UNIVERSITÄT DRESDEN
UNIVERSITÄT LEIPZIG

Introduction

In the life-sciences,

- many people need to write code for analysing data.
- They may not have coding skills.
- They use ChatGPT etc to generate code.
- The risk is that they use code that does wrong things.

- **Question:** How good are large language models (LLMs) e.g. for code generation specifically to analyse microscopy image data?

Introduction: Code generation using LLMs

- Codex model (Chen et al 2021) was major milestone in the code generation context – it's the predecessor of Github Copilot
- Chen et al 2021 also delivered a common benchmark: HumanEval

```
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]
```

Prompt

Reference solution

Figure 2. Three example problems from the HumanEval dataset, where the probabilities that a single sample from Codex-12B passes unit tests are 0.9, 0.17, and 0.005. The prompt provided to the model is shown with a white background, and a successful model-generated completion is shown in a yellow background. Though not a guarantee for problem novelty, all problems were hand-written and not programmatically copied from existing sources. Random problems and samples can be found in Appendix B.

Figure adapted from Chen et al. 2021
https://arxiv.org/abs/2107.03374

Benchmarking LLMs for Bio-image Analysis

Preprint

bioRχiv
THE PREPRINT SERVER FOR BIOLOGY

HOME | SUBMIT | FAQ | BLOG | ALERTS / RSS | RESOURCES | ABOUT | CHANNELS

Search
Advanced Search

New Results
Follow this preprint

Benchmarking Large Language Models for Bio-Image Analysis Code Generation

Robert Haase, Christian Tischer, Jean-Karim Hériché, Nico Scherf
doi: https://doi.org/10.1101/2024.04.19.590278
This article is a preprint and has not been certified by peer review [what does this mean?].

Abstract | Full Text | Info/History | Metrics
Preview PDF

Abstract

In the computational age, life-scientists often have to write Python code to solve bio-image analysis (BIA) problems. Many of them have not been formally trained in programming though. Code-generation, or coding assistance in general, with Large Language Models (LLMs) can have a clear impact on BIA. To the best of our knowledge, the quality of the generated code in this domain has not been studied. We present a quantitative benchmark to estimate the capability of LLMs to generate code for solving common BIA tasks. Our benchmark currently consists of 57 human-written prompts with corresponding reference solutions in Python, and unit-tests to evaluate functional correctness of potential solutions. We demonstrate our benchmark here and compare 18 state-of-the-art LLMs. To ensure that we will cover most of our community needs we also outline mid- and long-term strategies to maintain and extend the benchmark by the BIA open-source community. This work should support users in deciding for an LLM and also guide LLM developers in improving the capabilities of LLMs in the BIA domain.

Competing Interest Statement

The authors have declared no competing interest.

Copyright
The copyright holder for this preprint is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY 4.0 International license.

github.com/haesleinhuepf/human-eval-bia?tab=readme-ov-file

haesleinhuepf / human-eval-bia

Code | Issues 53 | Pull requests 12 | Discussions | Actions | Projects | Security | Insights | Settings

human-eval-bia Public
Unpin | Unwatch 3 | Fork 14 | Starred 23

main | 31 Branches | 4 Tags | Go to file | Add file | Code

haesleinhuepf Merge pull request #143 from haesleinhuepf/benchmark_sonnet_20241... 6157140 · 5 months ago 346 Commits

.github	fix typo	last year
data	evaluated claude-3-5-sonnet-20241022	6 months ago
demo	redraw figures	6 months ago

About
Benchmarking Large Language Models for Bio-Image Analysis Code Generation
python | benchmarking | bioimage-analysis | llm
Readme
MIT license

README | Code of conduct | MIT license

Benchmarking Large Language Models for Bio-Image Analysis Code Generation

DOI 10.5281/zenodo.12656870

This is a fork of the HumanEval repository where modifications were made to adapt the evaluation for Benchmarking LLMs in the Bio-image Analysis domain. The original HumanEval repository is an evaluation harness for the HumanEval problem solving dataset described in the paper "Evaluating Large Language Models Trained on Code".

Insights

Using the benchmark in this repository, we compared 15 LLMs in their capabilities to generate Bio-image Analysis Python code. Therefore, we defined test-cases listed here which can be used to evaluate functional correctness of bio-image analysis code. The pass-rate shown in this plot expresses the probability that generated code passed given unit-tests:

Suggested workflows
Based on your tech stack

- Python Package using Anaconda — Configure
  Create and test a Python package on multiple Python versions using Anaconda for package management.
- Publish Python Package — Configure
  Publish a Python Package to PyPI on release.
- SLSA Generic generator — Configure
  Generate SLSA3 provenance for your existing release workflows

More workflows | Dismiss suggestions

Code

Methods: test cases

Example test-case inspired by HumaEval (Chen et al 2021)

[1]:
```python
def workflow_segmentation_measurement_summary(image):
    """
    This function implements a workflow consisting of these steps:
    * threshold intensity input image using Otsu's method
    * label connected components
    * measure area of the labeled objects
    * determine mean area of all objects
    """
    import skimage
    import numpy as np
    binary_image = image > skimage.filters.threshold_otsu(image)
    label_image = skimage.measure.label(binary_image)
    stats = skimage.measure.regionprops(label_image)
    areas = [s.area for s in stats]
    return np.mean(areas)
```

Prompt

Reference solution

[2]:
```python
def check(candidate):
    import numpy as np

    assert candidate(np.asarray([
        [0,0,0,0,0],
        [1,1,1,0,0],
        [1,1,1,0,0],
        [1,1,0,0,0],
        [0,0,0,0,0],
    ])) == 8
```

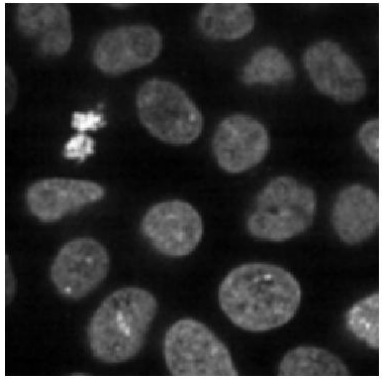
Unit test (excerpt)

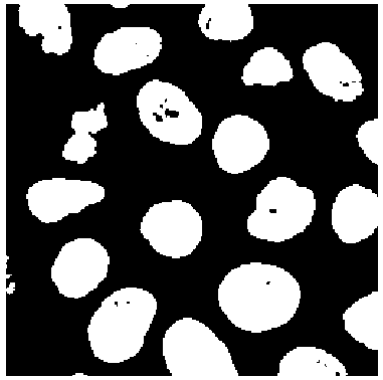
Unit test pass rate -> functional correctness

Chen at al 2021: https://arxiv.org/abs/2107.03374
https://www.biorxiv.org/content/10.1101/2024.04.19.590278v1
https://github.com/haesleinhuepf/human-eval-bia

Methods: test cases

Use case: segment the image and measure the average area of objects.





	area
0	955.0
1	31.0
2	815.0
3	1166.0
4	1135.0
5	680.0
6	1374.0
7	712.0
⋮	

858.04

Unit-test pass-rate (n=10):

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
workflow_segmentation_measurement_summary	1.0	0.9	1.0	0.8	0.5	0.5	0.1

https://www.biorxiv.org/content/10.1101/2024.04.19.590278v1
https://github.com/haesleinhuepf/human-eval-bia

Methods: test cases

Use-case: compute the correlation matrix

	a	b	c	d	e
0	1.600000	0.100000	1.600000	1.700000	1.700000
1	2.300000	0.200000	2.300000	2.400000	2.400000
2	2.600000	0.300000	2.600000	2.400000	2.400000
3	3.700000	0.300000	3.700000	3.600000	3.600000
4	3.400000	0.400000	3.400000	3.500000	3.500000
5	3.900000	0.400000	3.900000	3.900000	3.900000
6	4.300000	0.400000	4.300000	4.400000	4.400000
7	4.300000	0.500000	4.300000	4.200000	4.200000
8	4.000000	0.500000	4.000000	4.100000	4.100000
9	5.100000	0.500000	5.100000	5.000000	5.000000
10	5.200000	0.600000	5.200000	5.100000	5.100000
11	5.300000	0.600000	5.300000	5.400000	5.400000
12	5.500000	0.600000	5.400000	5.600000	5.600000

	a	b	c	d	e
a	1.000000	0.949504	0.999775	0.995800	0.995800
b	0.949504	1.000000	0.949594	0.946039	0.946039
c	0.999775	0.949594	1.000000	0.995001	0.995001
d	0.995800	0.946039	0.995001	1.000000	1.000000
e	0.995800	0.946039	0.995001	1.000000	1.000000

Unit-test pass-rate (n=10):

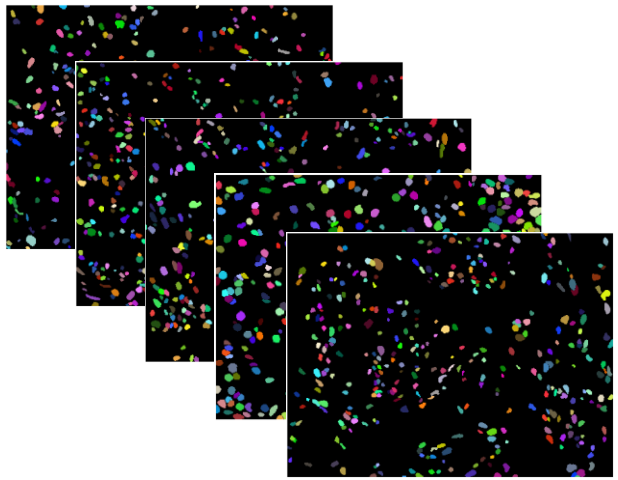
	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
pair_wise_correlation_matrix	1.0	1.0	1.0	0.9	1.0	0.5	0.1

https://www.biorxiv.org/content/10.1101/2024.04.19.590278v1
https://github.com/haesleinhuepf/human-eval-bia

Methods: test cases

Use case: Count segmented objects in a folder of segmentation results.

Ganglioneuroblastoma_0.tif
Ganglioneuroblastoma_1.tif
Ganglioneuroblastoma_2.tif
Ganglioneuroblastoma_3.tif
Ganglioneuroblastoma_4.tif



300
398
368
378
363

Unit-test pass-rate (n=10):

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
workflow_batch_process_folder_count_labels	1.0	0.1	0.0	0.3	0.0	0.0	0.0

https://www.biorxiv.org/content/10.1101/2024.04.19.590278v1
https://github.com/haesleinhuepf/human-eval-bia
Data Source: https://www.ebi.ac.uk/bioimage-archive/galleries/S-BIAD634-ai.html

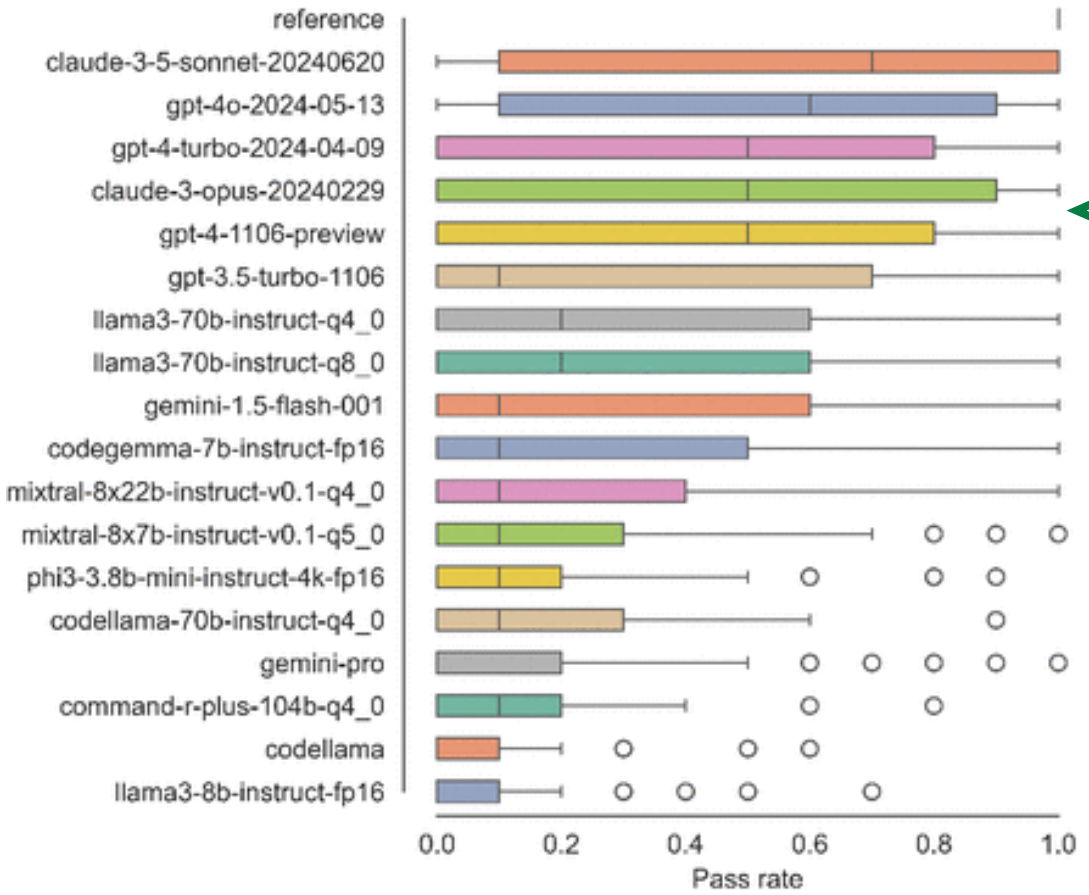
Results

Unit-test pass-rate (n=10)

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
Statistics / tabular data wrangling							
combine_columns_of_tables	1.0	0.8	0.1	1.0	0.9	0.7	0.1
create_umap	1.0	0.8	1.0	0.9	1.0	0.8	0.0
t_test	1.0	1.0	1.0	0.9	1.0	0.5	0.3
Measurements / feature extraction							
measure_intensity_over_time	1.0	0.9	0.4	0.1	0.4	0.0	0.1
measure_intensity_of_labels	1.0	0.2	0.4	0.4	0.1	0.0	0.0
measure_properties_of_regions	1.0	0.4	0.6	0.8	0.2	0.0	0.1
count_number_of_touching_neighbors	1.0	0.6	0.1	0.2	0.1	0.0	0.0
Advanced workflows / big data							
tiled_image_processing	1.0	0.2	0.0	0.0	0.0	0.0	0.0
workflow_batch_process_folder_measure_intensity	1.0	0.5	0.0	0.9	0.1	0.0	0.0

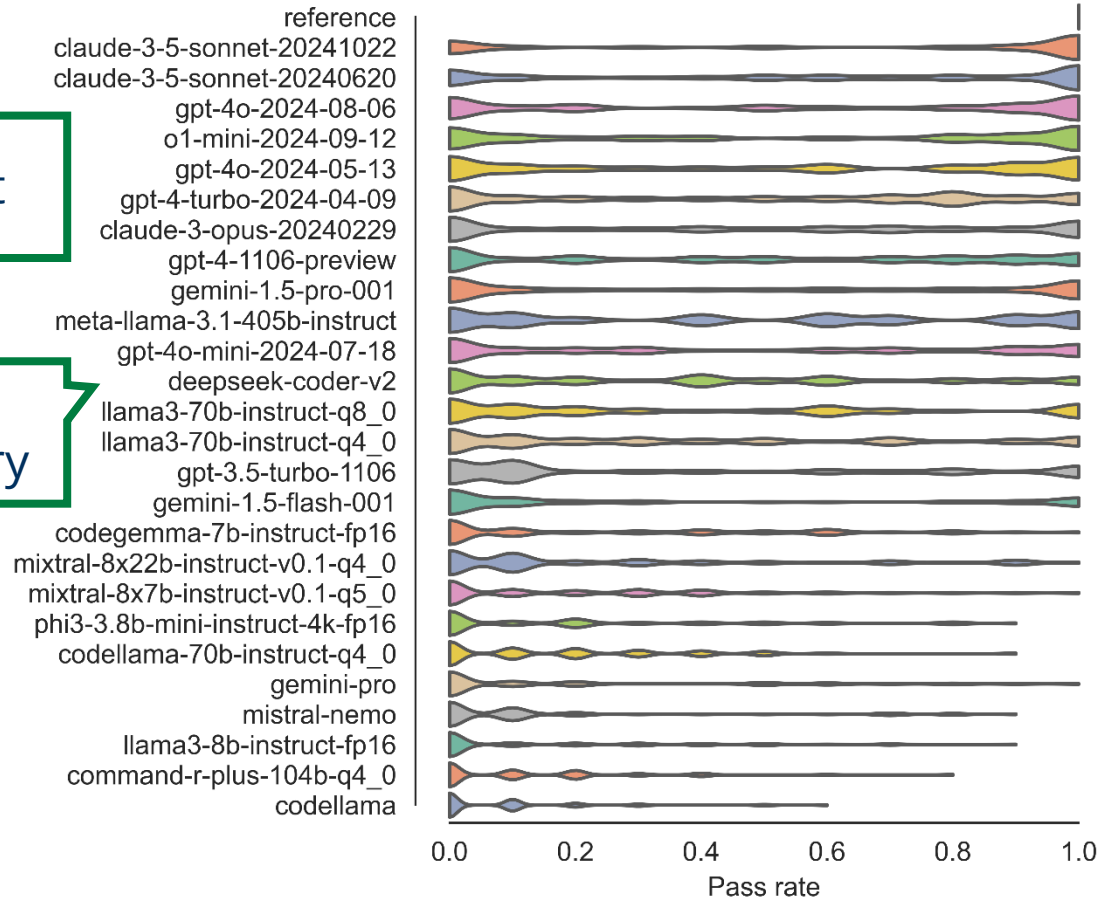
Results

Summary: 57 use-cases (yet), 26 LLMs (yet), n=10



Preprint

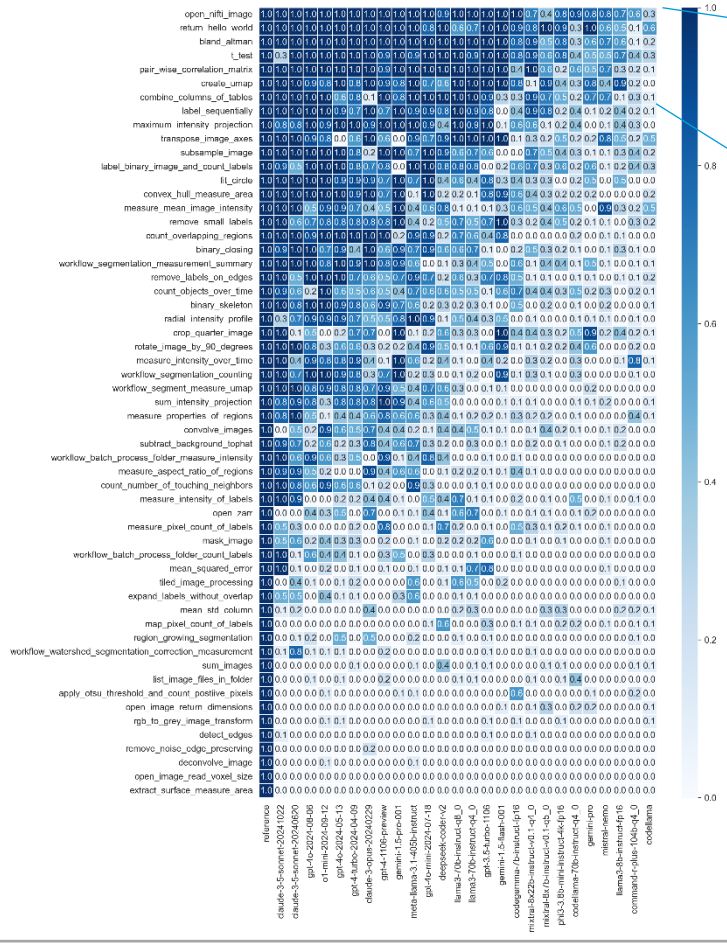
Github repository

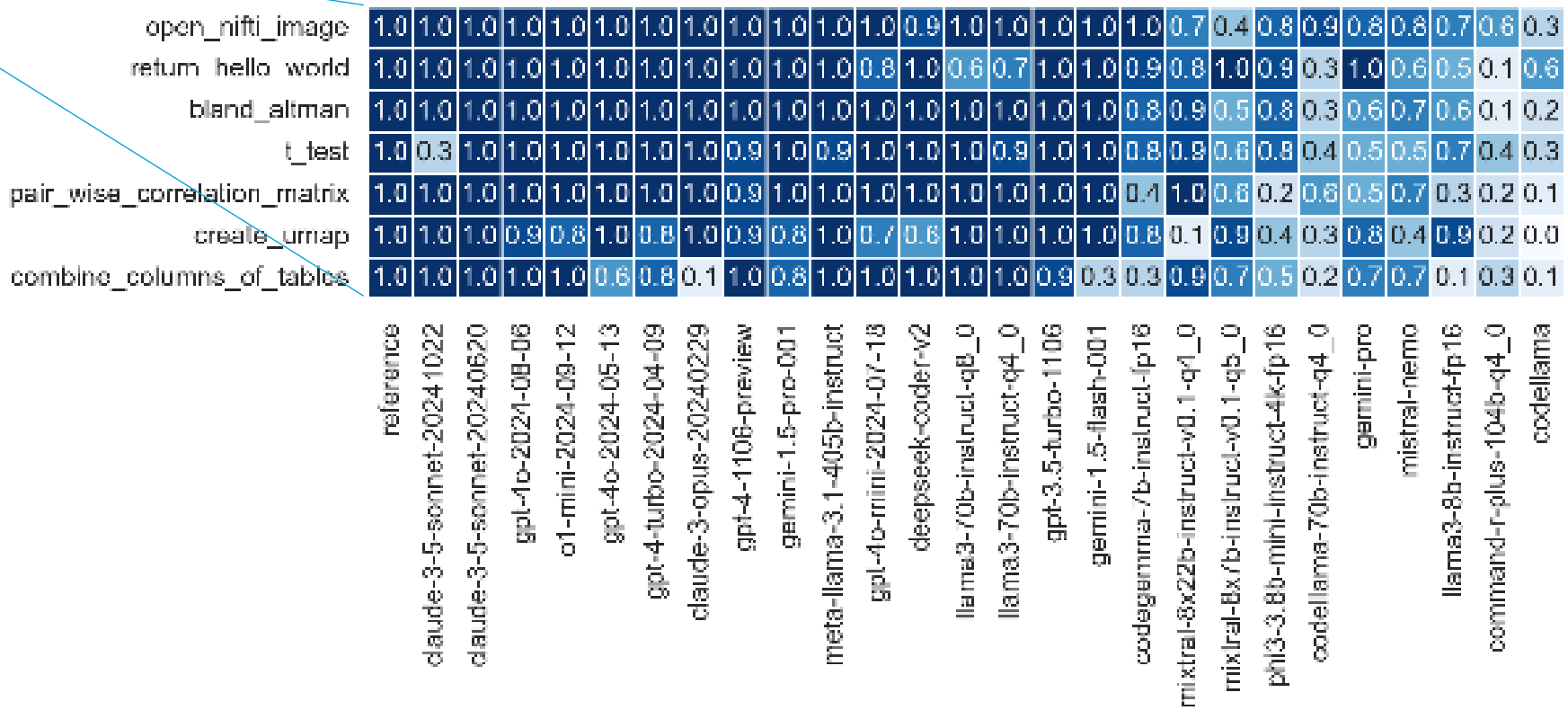


https://www.biorxiv.org/content/10.1101/2024.04.19.590278v3
https://github.com/haesleinhuepf/human-eval-bia/

Results in more detail

Summary: 57 use-cases (yet), 26 LLMs (yet), n=10

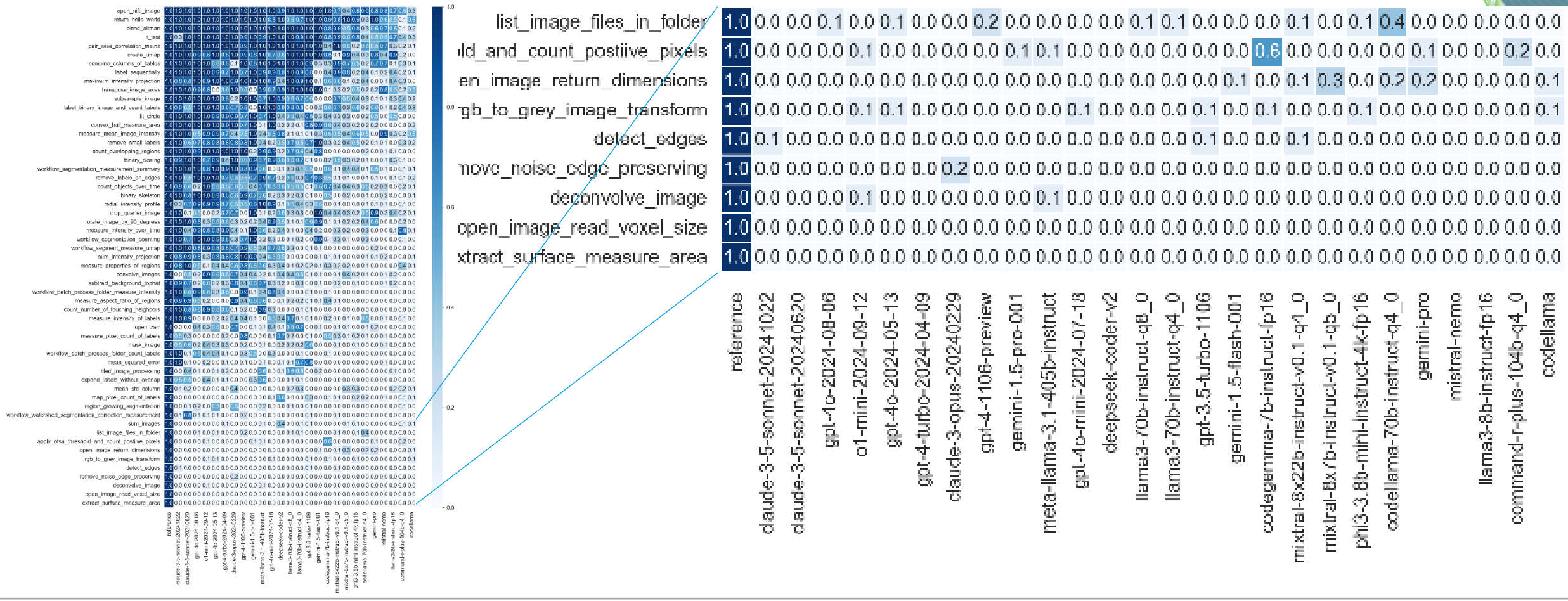




https://www.biorxiv.org/content/10.1101/2024.04.19.590278v3
https://github.com/haesleinhuepf/human-eval-bia/

Results in more detail

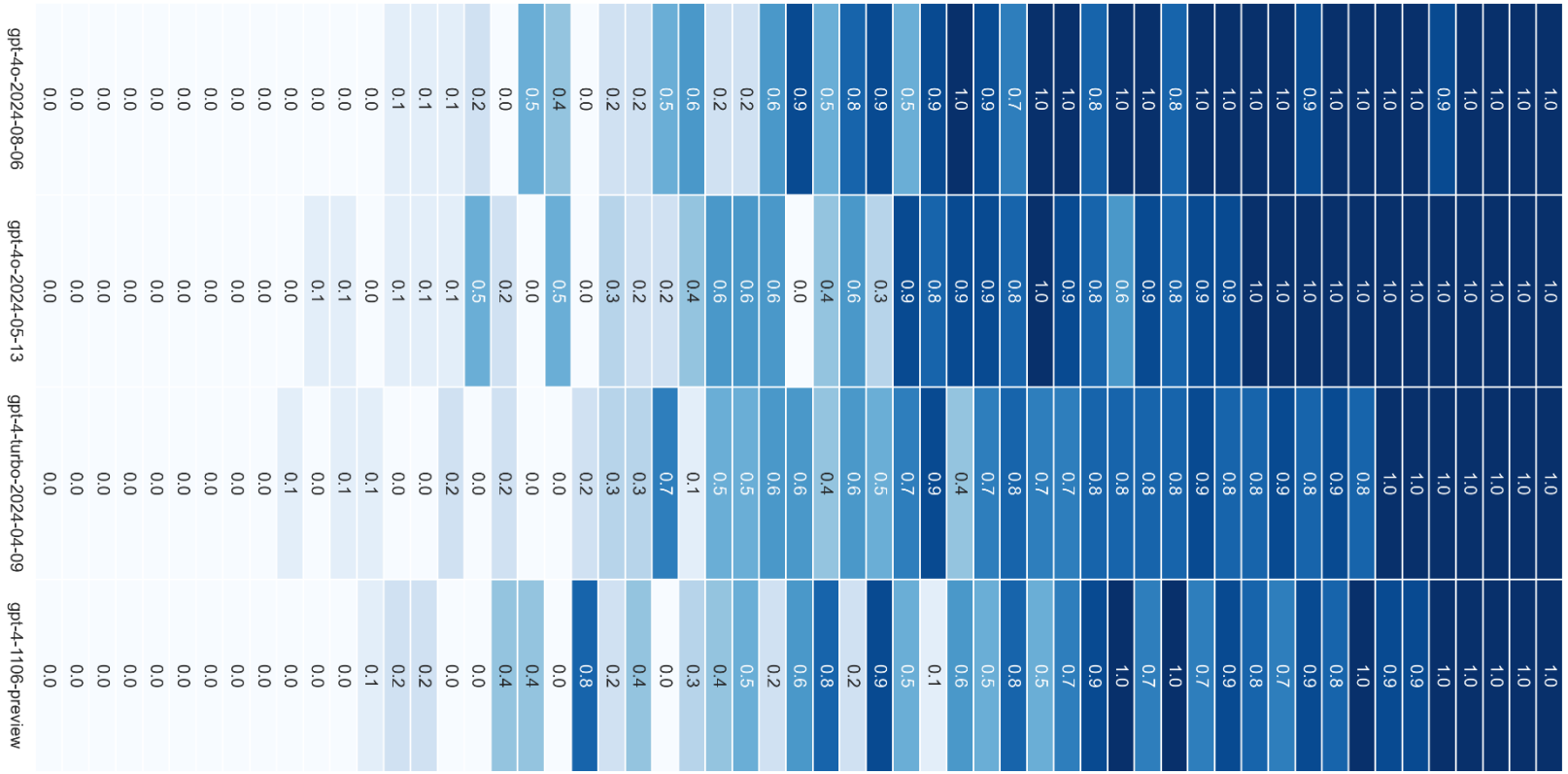
Summary: 57 use-cases (yet), 26 LLMs (yet), n=10



https://www.biorxiv.org/content/10.1101/2024.04.19.590278v3
https://github.com/haesleinhuepf/human-eval-bia/

Results in even more detail

Gpt4 improvement over 10 months: 7%



gpt 4o (August 2024)	53%
gpt 4o (May 2024)	51%
gpt 4turbo (April 2024)	47%
gpt 4preview (November 2023)	46%

Additional analysis, not online

Results: strengths and weaknesses of LLMs

Common error messages, e.g. for different versions of the gemini model (Google)

Missing import statements

	claude-3-5-sonnet-20241022	claude-3-5-sonnet-20240620	gpt-4o-2024-08-06	o1-mini-2024-09-12	gpt-4o-2024-05-13	gpt-4-turbo-2024-04-09	claude-3-opus-20240229	gpt-4-1106-preview	gemini-1.5-pro-001	meta-llama-3.1-405b-instruct	gpt-4o-mini-2024-07-18	deepseek-coder-v2	llama3-70b-instruct-q8_0	llama3-70b-instruct-q4_0	gpt-3.5-turbo-1106	gemini-1.5-flash-001	codegemma-7b-instruct-fp16	mixtral-8x22b-instruct-v0.1-q4_0	mixtral-8x7b-instruct-v0.1-q5_0	phi3-3.8b-mini-instruct-4k-fp16	codellama-70b-instruct-q4_0	gemini-pro	mistral-nemo	llama3-8b-instruct-fp16	command-r-plus-104b-q4_0	codellama
has no attribute	25	28	32	30	33	33	45	48	35	46	43	31	38	42	37	28	49	58	54	44	41	43	59	60	25	59
invalid syntax	0	0	0	0	0	0	0	1	0	0	2	4	5	6	4	1	1	3	7	30	56	0	24	10	167	58
is not defined	1	0	0	0	0	4	5	8	10	3	4	2	1	3	11	35	86	5	27	26	36	203	28	12	45	32
Can't convert object	0	0	0	0	0	1	1	3	0	0	0	0	0	0	9	0	1	6	3	12	13	3	0	0	0	13
cannot import	0	0	3	2	2	3	5	2	11	4	0	7	12	2	2	13	15	4	7	6	7	6	19	12	6	17
out of range	0	0	0	0	1	0	3	0	11	3	0	0	1	0	0	20	5	1	9	4	4	0	4	3	1	4
unexpected keyword argument	8	8	8	7	5	15	5	8	9	4	4	11	10	5	7	2	8	3	13	8	8	1	9	4	0	4

https://www.biorxiv.org/content/10.1101/2024.04.19.590278v3
https://github.com/haesleinhuepf/human-eval-bia/

Results: strengths and weaknesses of LLMs

- LLMs use different Python libraries than we Bio-image Analysts do.
- What can we teach LLMs?
- What can we learn from this?

	reference	claude-3-5-sonnet-20241022	claude-3-5-sonnet-20240620	gpt-4o-2024-08-06	o1-mini-2024-09-12	gpt-4o-2024-05-13	gpt-4-turbo-2024-04-09	claude-3-opus-20240229	gpt-4-1106-preview	gemini-1.5-pro-001	meta-llama-3.1-405b-instruct	gpt-4o-mini-2024-07-18	deepseek-coder-v2	llama3-70b-instruct-q8_0	llama3-70b-instruct-q4_0	gpt-3.5-turbo-1106	gemini-1.5-flash-001	codegemma-7b-instruct-fp16	mixtral-8x22b-instruct-v0.1-q4_0	mixtral-8x7b-instruct-v0.1-q5_0	phi3-3.8b-mini-instruct-4k-fp16	codellama-70b-instruct-q4_0	gemini-pro	mistral-nemo	llama3-8b-instruct-fp16	command-r-plus-104b-q4_0	codellama
numpy	220	497	487	453	435	442	434	453	398	322	475	477	480	460	447	360	384	298	478	392	450	426	165	403	432	412	454
scipy	70	118	170	104	112	118	123	131	141	65	126	141	89	156	144	76	57	76	168	82	138	118	31	133	155	82	114
skimage	220	124	115	110	108	102	129	125	132	149	119	87	149	98	85	115	91	154	118	102	129	151	116	60	68	131	96
cv2	0	56	45	51	31	66	63	44	57	52	40	100	112	85	107	144	107	43	90	76	107	120	82	137	192	31	137
pandas	60	95	100	98	97	99	100	99	97	68	100	100	101	100	100	90	88	74	98	72	99	81	52	97	98	89	95
pyclesperanto_prototype	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
vedo	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
umap	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	16	20	20	20	19	20
dask	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
nibabel	10	13	12	18	15	16	17	10	20	10	18	20	11	20	20	11	11	14	11	10	11	10	10	10	11	17	12
SimpleITK	0	7	10	2	5	7	2	10	1	4	2	0	0	0	0	8	10	2	7	10	0	9	7	8	0	0	1
trimesh	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	2	1	1	0	0	0	1	0	0	0	0
itk	0	7	10	2	5	6	2	10	0	3	2	0	0	0	0	9	10	4	8	10	0	9	7	6	2	0	1

Human reference

https://www.biorxiv.org/content/10.1101/2024.04.19.590278v3
https://github.com/haesleinhuepf/human-eval-bia/

Additional online resources

Step-by-step guide for running the benchmark

haesleinhuepf/human-eval-bia

github.com/haesleinhuepf/human-eval-bia/tree/main

README | Code of conduct | MIT license

Installation

Make sure to use python 3.10 or later:

```
$ mamba create --name heb python=3.10
$ conda activate heb
```

Check out and install this repository:

```
$ git clone https://github.com/haesleinhuepf/human-eval-bia.git
$ cd human-eval-bia
$ pip install -e .
$ pip install -r requirements.txt
```

To run the benchmark for OpenAI-based models, please create an OpenAI API Key as explained in this blog post.

haesleinhuepf/human-eval-bia

github.com/haesleinhuepf/human-eval-bia/tree/main

README | Code of conduct | MIT license

To reproduce our benchmarks, you can go through the notebooks provided in the `/notebooks` directory:

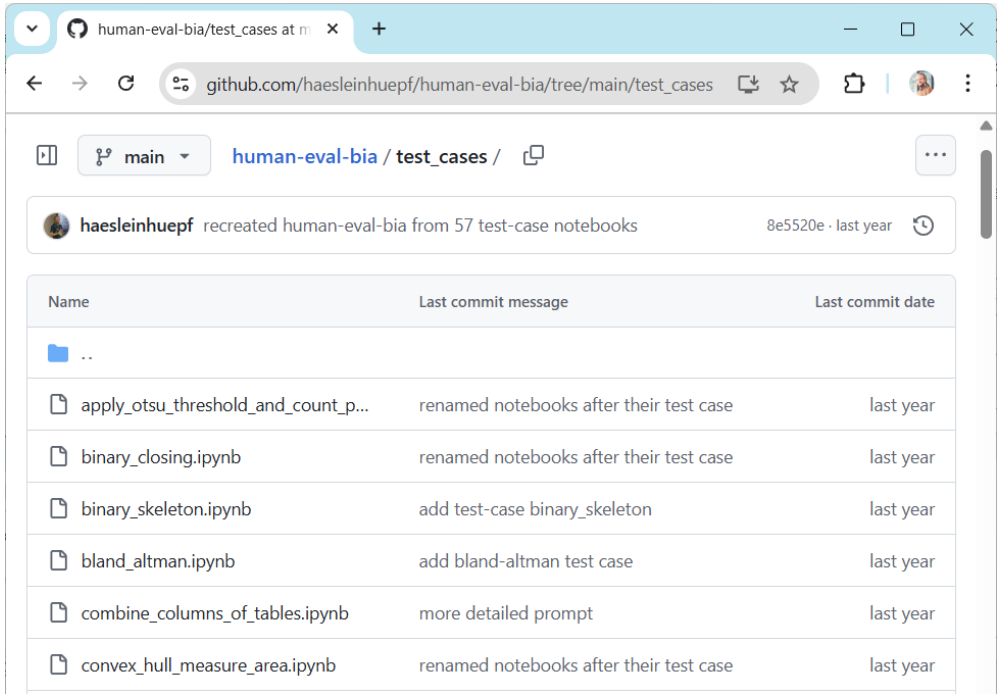
- Create benchmarking set from notebooks: Convert notebook files into HumanEval-compatible JSONL files.
- Generate code samples from LLMs: Send prompts to LLMs multiple times and store results in sample_....JSONL files.
- Evaluate samples by executing code: Run the code samples and store the results in _result....JSONL files.
- Detect missing requirements: In case this notebook lists missing requirements, that can be installed, the benchmark maintainers may add those requirements and run the evaluation step again.
- Summarize pass rates pass@k: Draw box-plot with pass-rates for all LLMs.
- Summarize pass rates grouped by test-cases: Visualize for each test-case the pass-rate of all LLMs individually.
- Summarize error messages: Count common error messages in evaluation responses.
- Summarize used Python libraries: Count common Python Libraries in evaluation

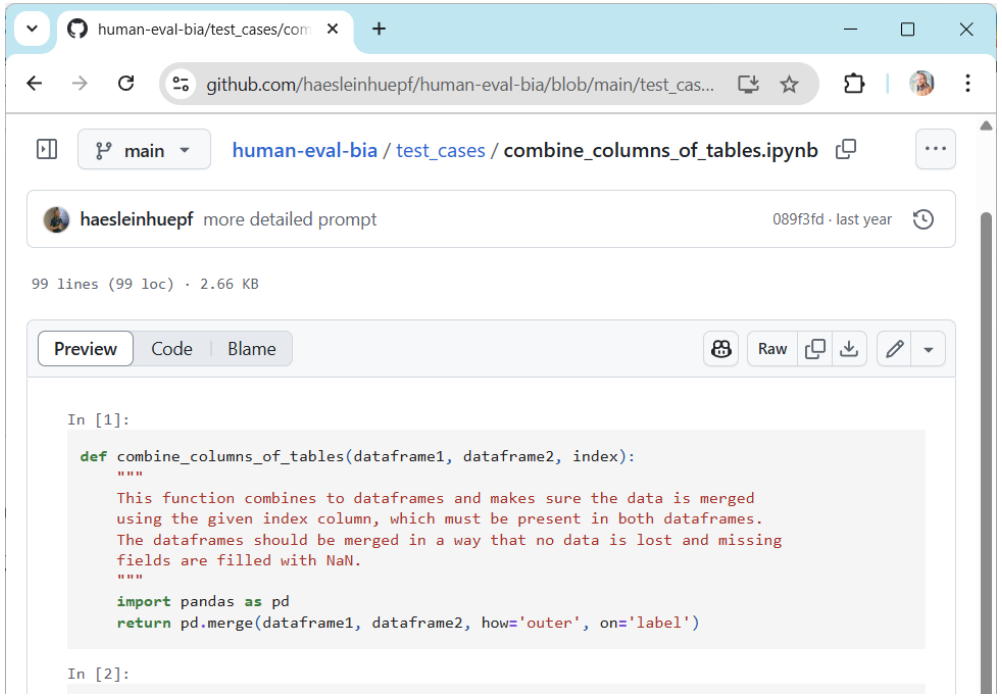
Additional online resources

Complete set of use-cases online, open source

Advantage: Reproducibility

Disadvantage: Enables cheating in model training

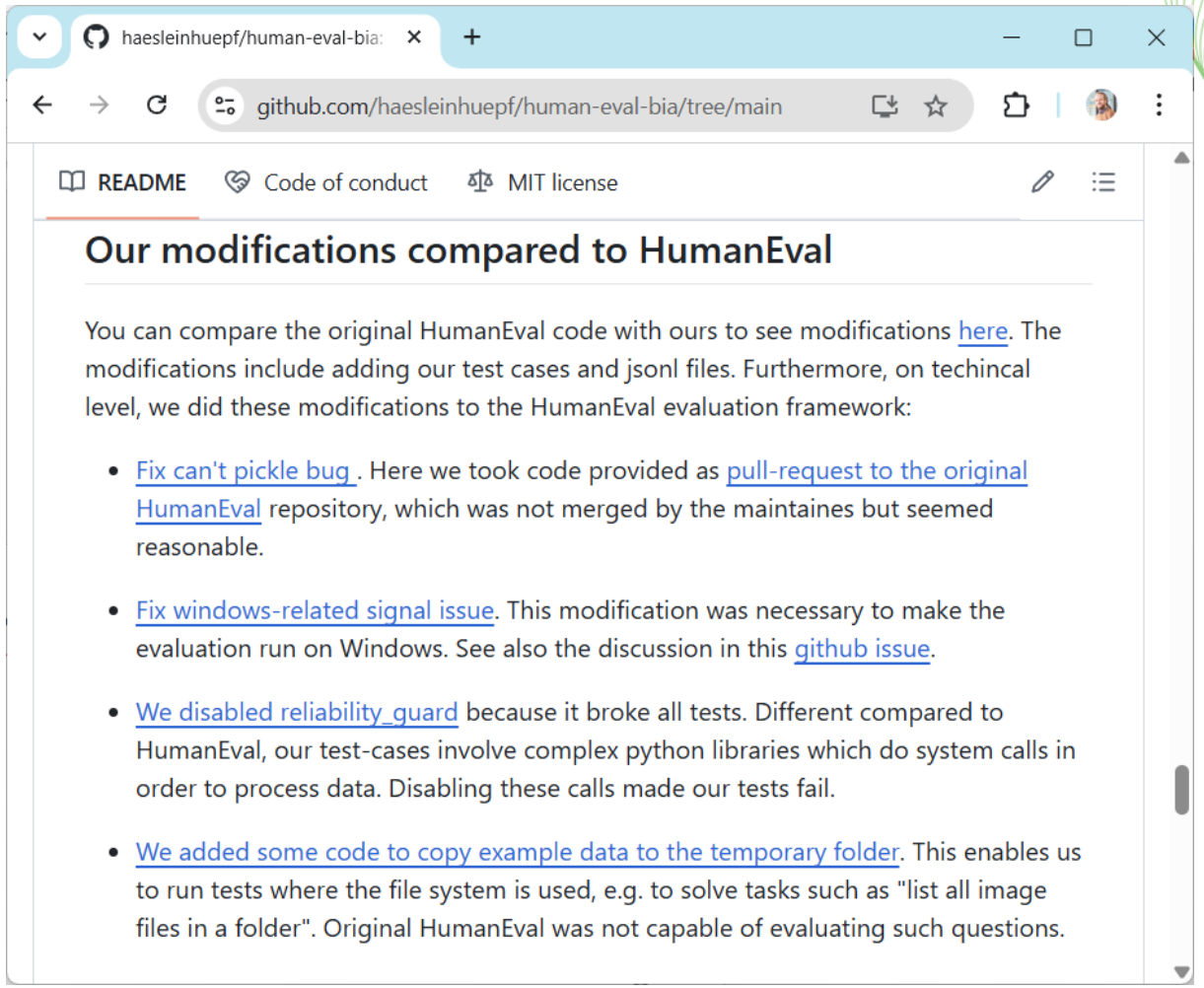




https://github.com/haesleinhuepf/human-eval-bia/tree/main/test_cases

HumanEval versus HumanEvalBIA

- HumanEval does not
  - work with image data
  - work with files
  - does not work on Windows
  - allow system calls
- The HumanEval authors did not publish
  - all test-cases (just a selection)
  - the code for creating the samples.



https://github.com/haesleinhuepf/human-eval-bia/tree/main?tab=readme-ov-file#our-modifications-compared-to-humaneval

Critics

- Benchmark only measures functional correctness of code.
- Functional correctness in relation to executability would be interesting, too. -> How often does code run, but produce wrong results?
- Potential extensions:
  - Measure performance (speed, memory consumption)
  - Code quality

Critics

- 57 test-cases are not many. (20 more in the making)
- Most test-cases written by a single person -> introduces bias
- Open benchmarks may become part of training data of future models; rendering the benchmark useless for benchmarking

Summary

- First benchmark of its kind in the bio-imaging / microscopy community
- Shows that commercial LLMs reach functional correctness of 50-60%
- Open-weight LLMs achieve < 50%
- May be useful for developing better models

Questions?