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

Large Language Models for Quantitative Bio-image Analysis

Robert Haase

https://doi.org/10.5281/zenodo.12518075

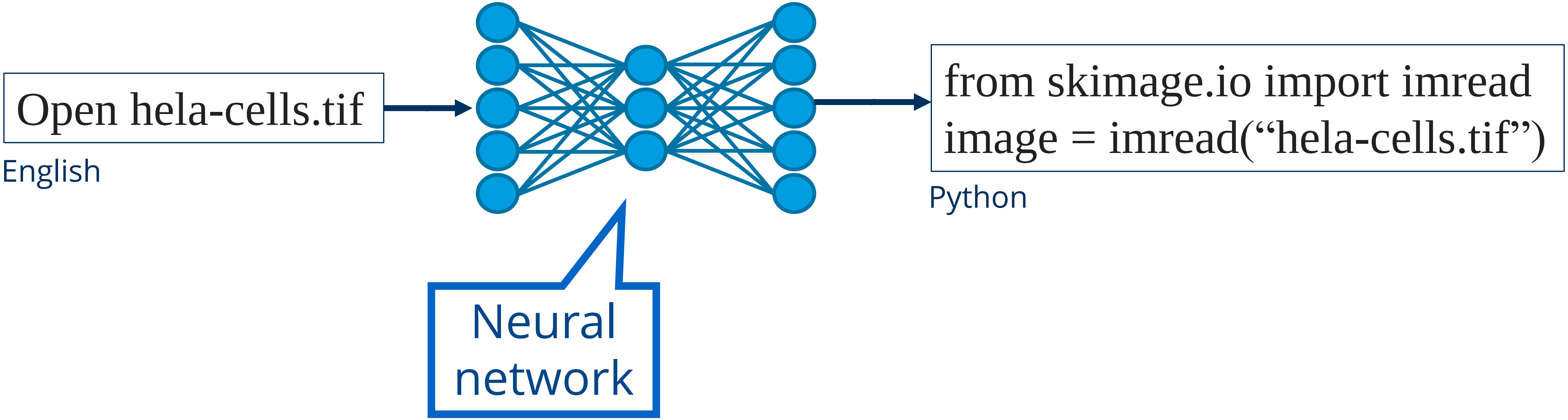
Bundesministerium
für Bildung
und Forschung

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.

Large Language Models (LLMs)

Text-to-text, translation, code generation



Open hela-cells.tif

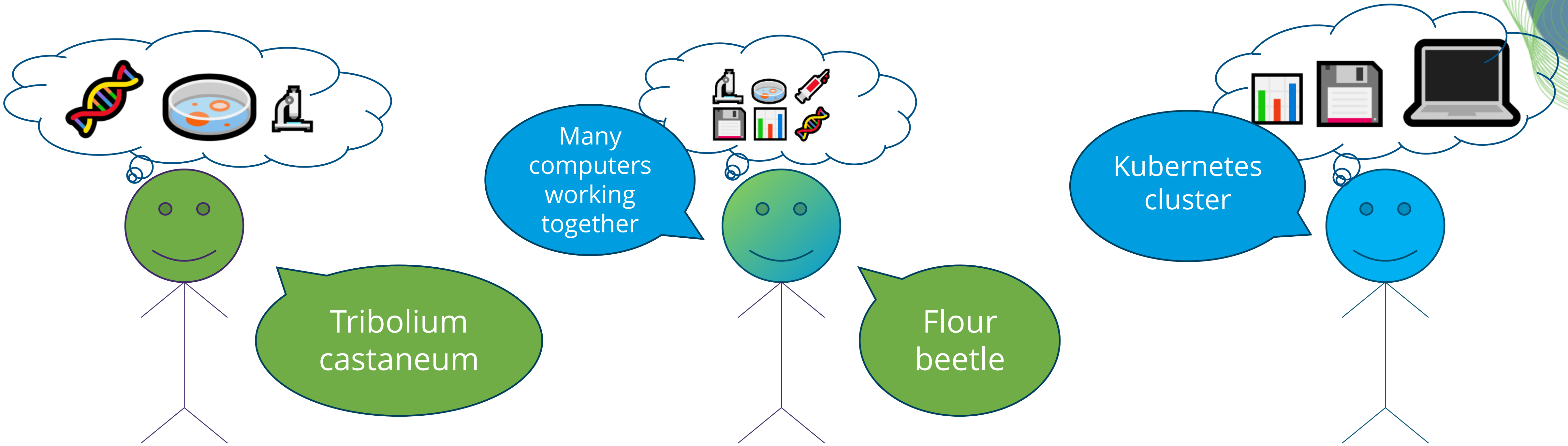
English

```python
from skimage.io import imread
image = imread(“hela-cells.tif”)
```

Python

Neural network

Bio-image Analysis



Biologist

Domain-specialist
(focused on
real-world problems)

Bio-image Analyst

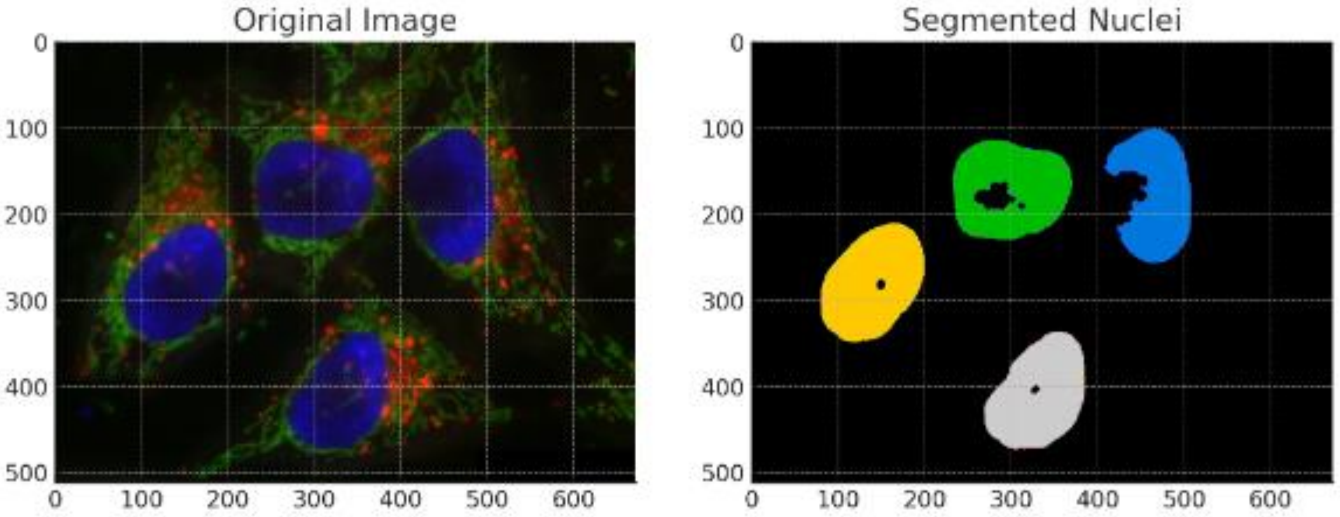
← Generalist →
(data-driven,
service-oriented)

Computer Scientist

Method + infrastructure specialist
(algorithm-centered)

Bio-image Analysis

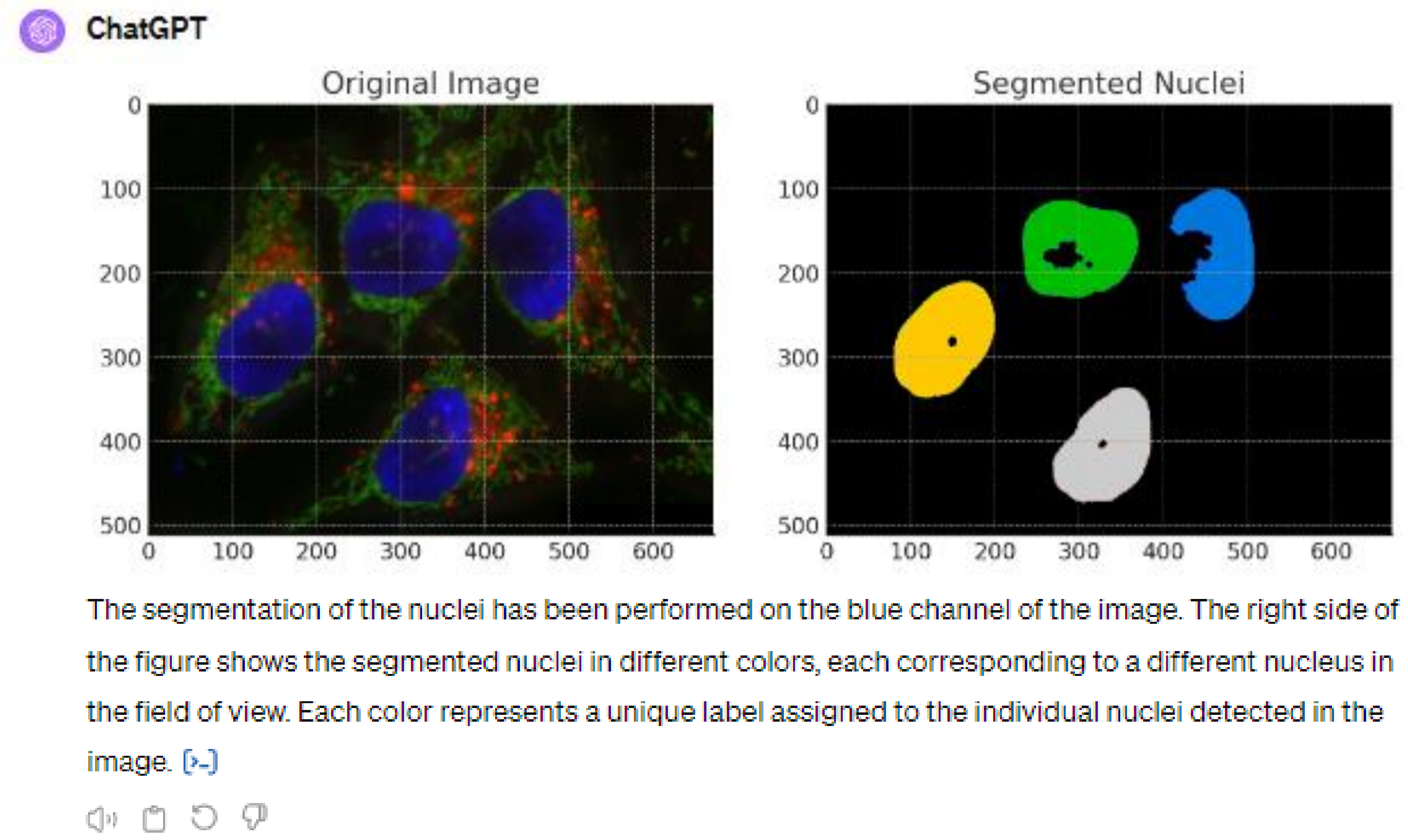
My job ...



Bio-image Analysis using Large-Language Models

My job is changing, since we have ChatGPT

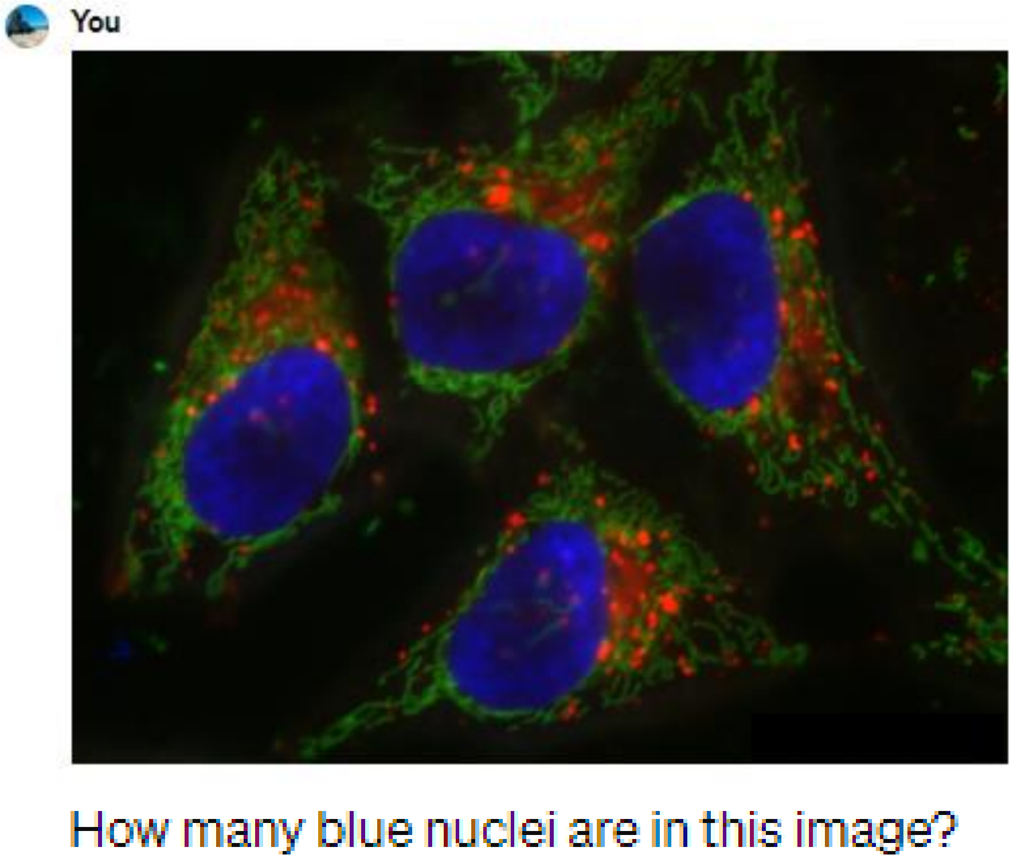
Prompt:
“Please segment the nuclei in the blue channel of the image”



The segmentation of the nuclei has been performed on the blue channel of the image. The right side of the figure shows the segmented nuclei in different colors, each corresponding to a different nucleus in the field of view. Each color represents a unique label assigned to the individual nuclei detected in the image. [>_]

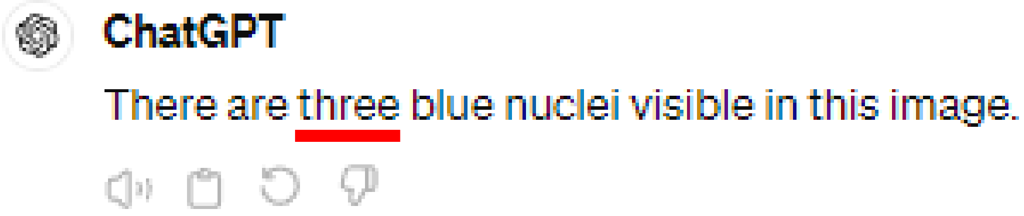
Vision language models for counting objects

LLMs were not exactly built for this use-case…



You

How many blue nuclei are in this image?



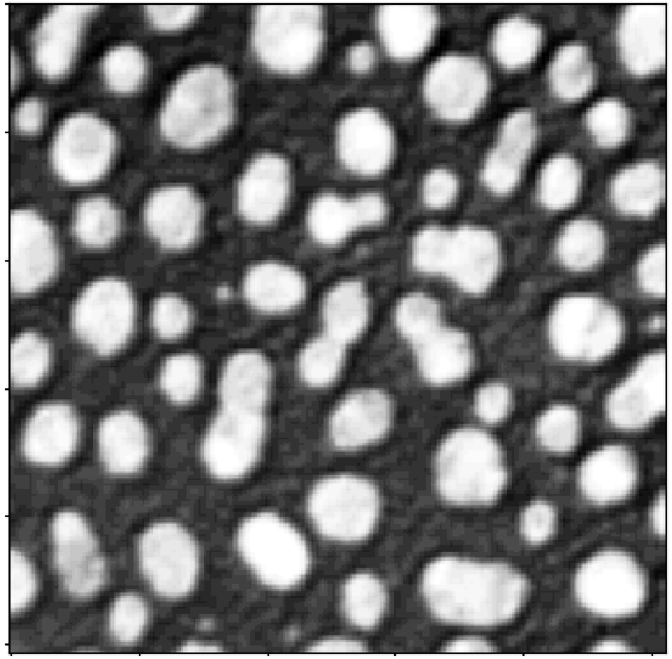
ChatGPT

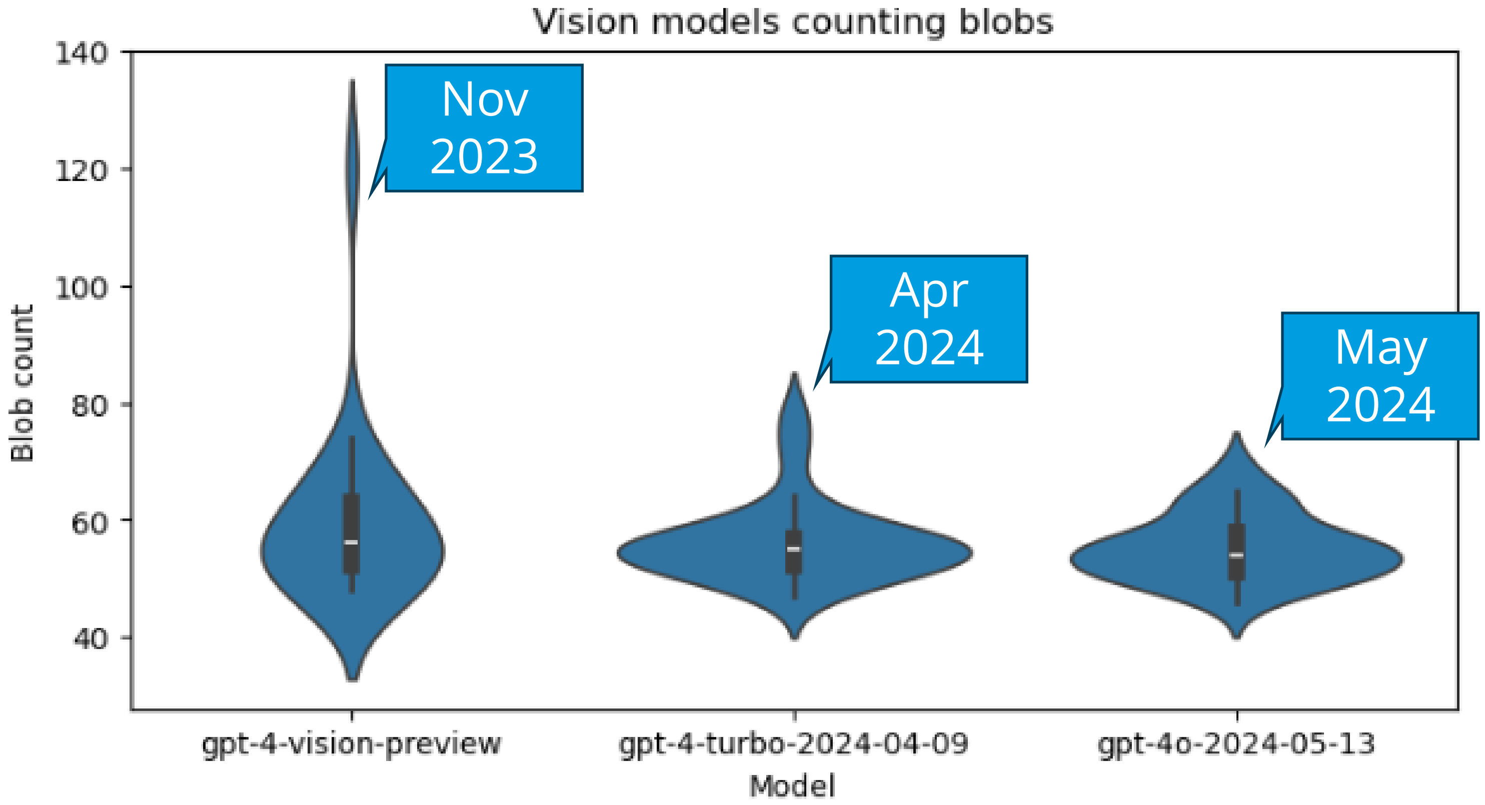
There are three blue nuclei visible in this image.

n=1

Vision language models for counting objects

Prompt: „Analyse the following image by counting the bright blobs. Respond with the number only. “ (n=25)





Nov 2023

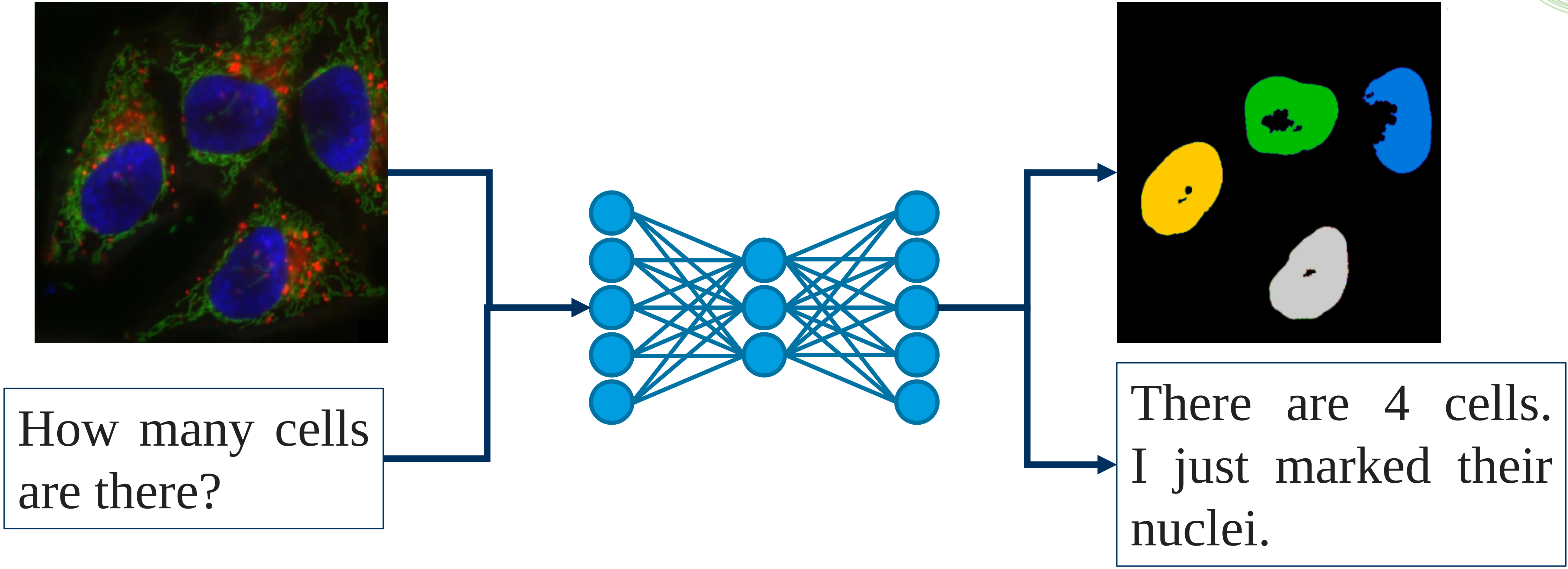
Apr 2024

May 2024

https://haesleinhuepf.github.io/BioImageAnalysisNotebooks/20c_vision_models/vision_models.html

Multi-modal LLMs

Combining image, text and […] data, to gain new [biological] insights.



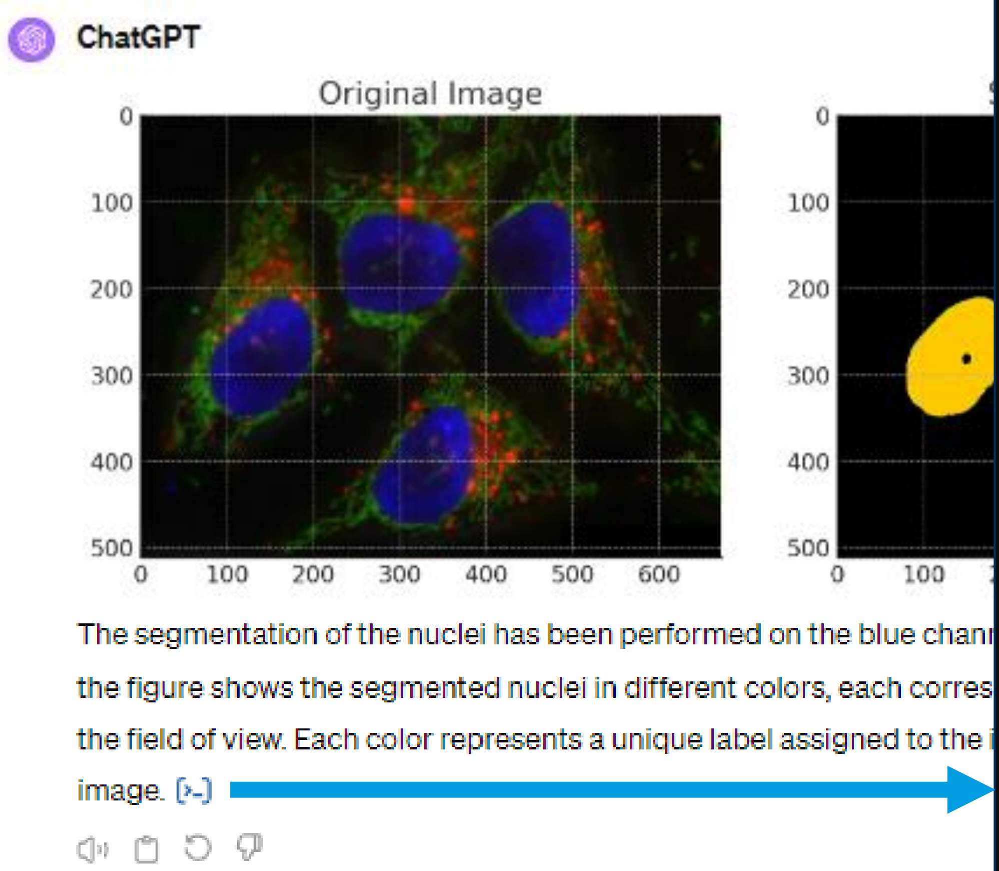
How many cells are there?

There are 4 cells. I just marked their nuclei.

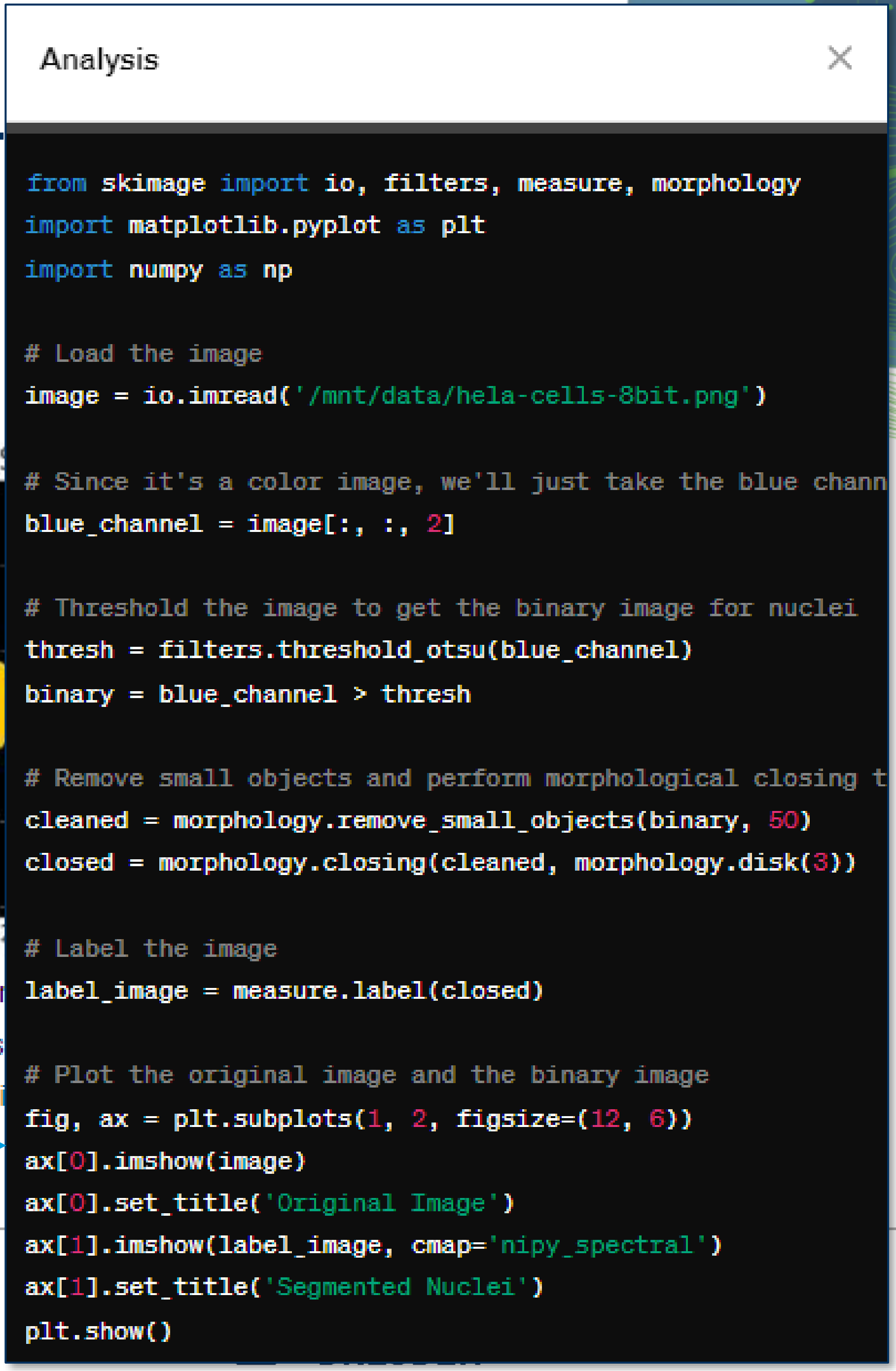
Bio-image Analysis using Large-L

My job is changing, since we have ChatGPT

Prompt:
“Please segment the nuclei in the blue channel of the image”



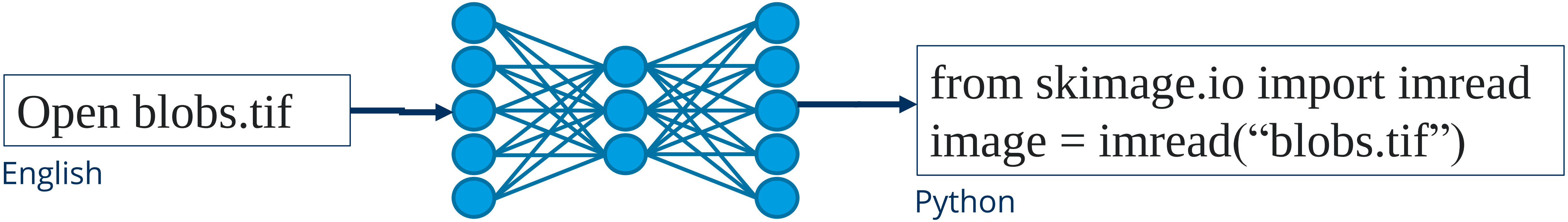
The segmentation of the nuclei has been performed on the blue chann
the figure shows the segmented nuclei in different colors, each corres
the field of view. Each color represents a unique label assigned to the i
image. [>_]



```
from skimage import io, filters, measure, morphology
import matplotlib.pyplot as plt
import numpy as np

# Load the image
image = io.imread('/mnt/data/hela-cells-8bit.png')

# Since it's a color image, we'll just take the blue chann
blue_channel = image[:, :, 2]

# Threshold the image to get the binary image for nuclei
thresh = filters.threshold_otsu(blue_channel)
binary = blue_channel > thresh

# Remove small objects and perform morphological closing t
cleaned = morphology.remove_small_objects(binary, 50)
closed = morphology.closing(cleaned, morphology.disk(3))

# Label the image
label_image = measure.label(closed)

# Plot the original image and the binary image
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
ax[0].imshow(image)
ax[0].set_title('Original Image')
ax[1].imshow(label_image, cmap='nipy_spectral')
ax[1].set_title('Segmented Nuclei')
plt.show()
```

Large Language Models (LLMs)

Text-to-text, translation, code generation



Open blobs.tif

English

```
from skimage.io import imread
image = imread(“blobs.tif”)
```

Python

Benchmarking LLMs for Bio-image Analysis

Which LLM should I pay for?

Is GPT-4o better than Claude?

Do LLMs introduce bias?

How can we validate LLMs for BiA?

How *good* are LLMs for Bio-image Analysis?

Can LLMs act according to good scientific practice?

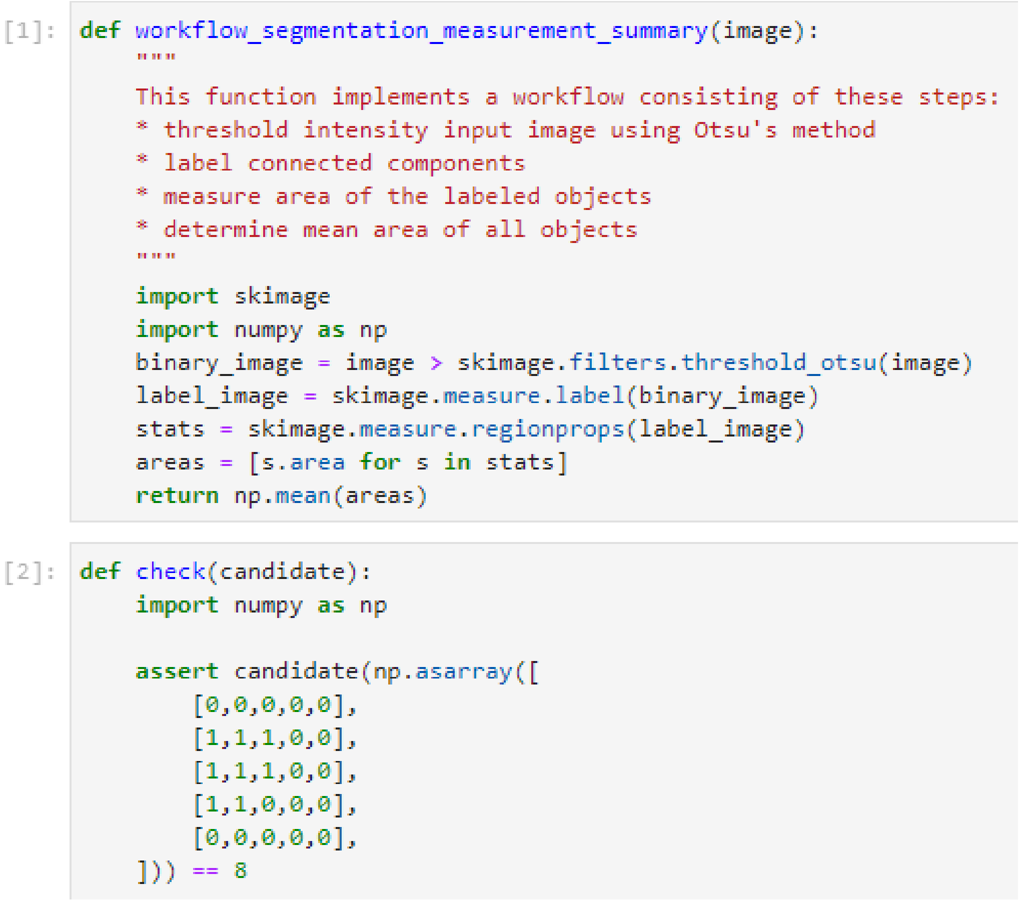
Will we all be replaced by AI anytime soon?

What are the limits of the LLM technology?

How can we validate specific LLM output without ground truth?

Benchmarking LLMs for Bio-image Analysis

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)
```

[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
```

Prompt

Reference solution

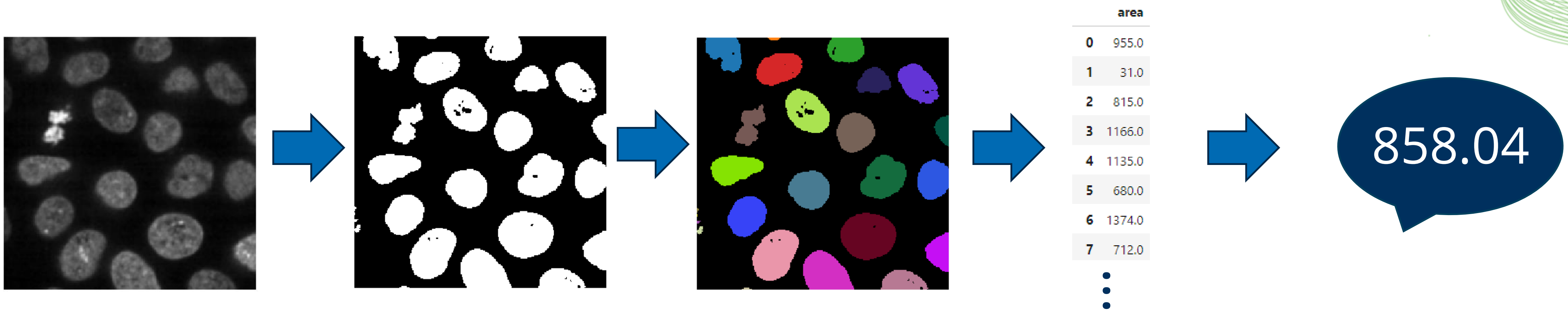
Unit test (excerpt)

We formulated 57 of such test-cases (yet)

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

Benchmarking LLMs for Bio-image Analysis

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

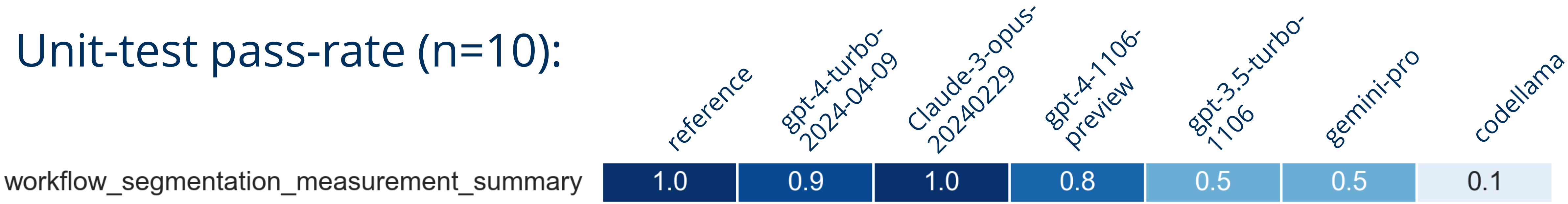


	area
0	955.0
1	31.0
2	815.0
3	1166.0
4	1135.0
5	680.0
6	1374.0
7	712.0

⋮

858.04

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

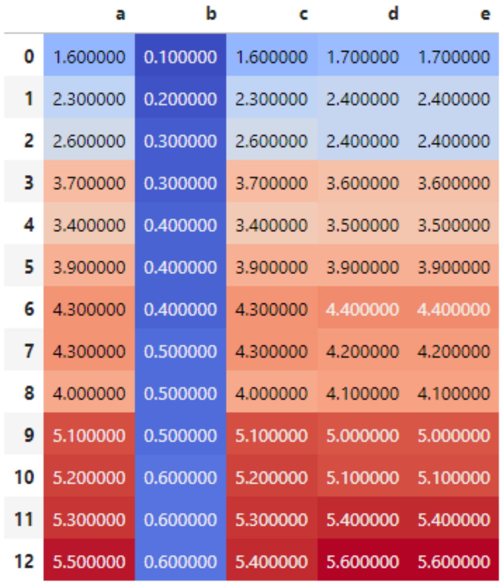


	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
workflow_segmentation_measurement_summary	1.0	0.9	1.0	0.8	0.5	0.5	0.1

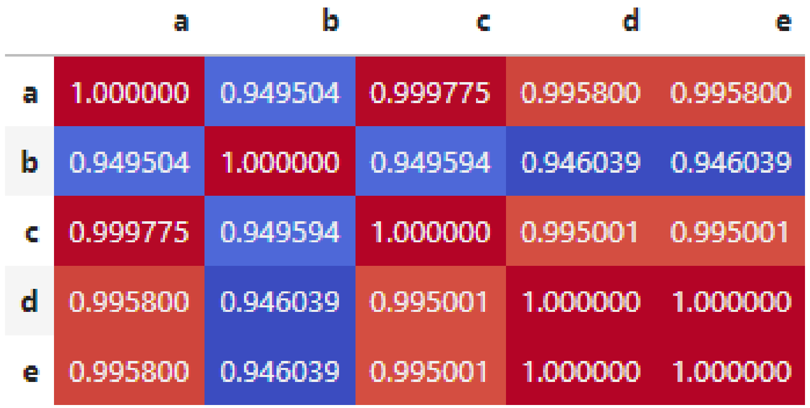
https://www.biorxiv.org/content/10.1101/2024.04.19.590278v1
https://github.com/haesleinhuepf/human-eval-bia

Benchmarking LLMs for Bio-image Analysis

Use-case: compute the correlation matrix

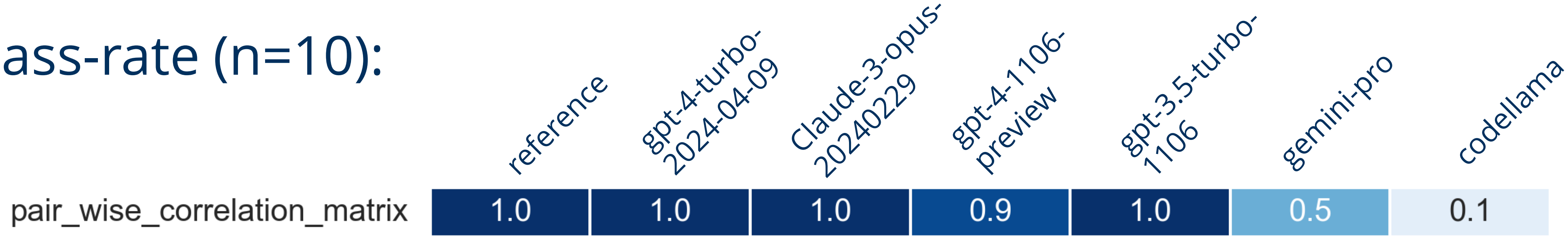


	a	b	c	d	e
0	1.600000	0.100000	1.600000	1.700000	1.700000
1	2.300000	0.200000	2.300000	2.400000	2.400000
2	2.600000	0.300000	2.600000	2.400000	2.400000
3	3.700000	0.300000	3.700000	3.600000	3.600000
4	3.400000	0.400000	3.400000	3.500000	3.500000
5	3.900000	0.400000	3.900000	3.900000	3.900000
6	4.300000	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):



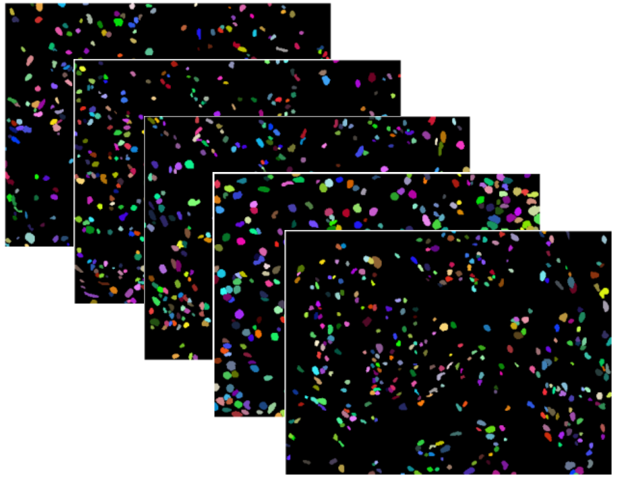
	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
pair_wise_correlation_matrix	1.0	1.0	1.0	0.9	1.0	0.5	0.1

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

Benchmarking LLMs for Bio-image Analysis

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



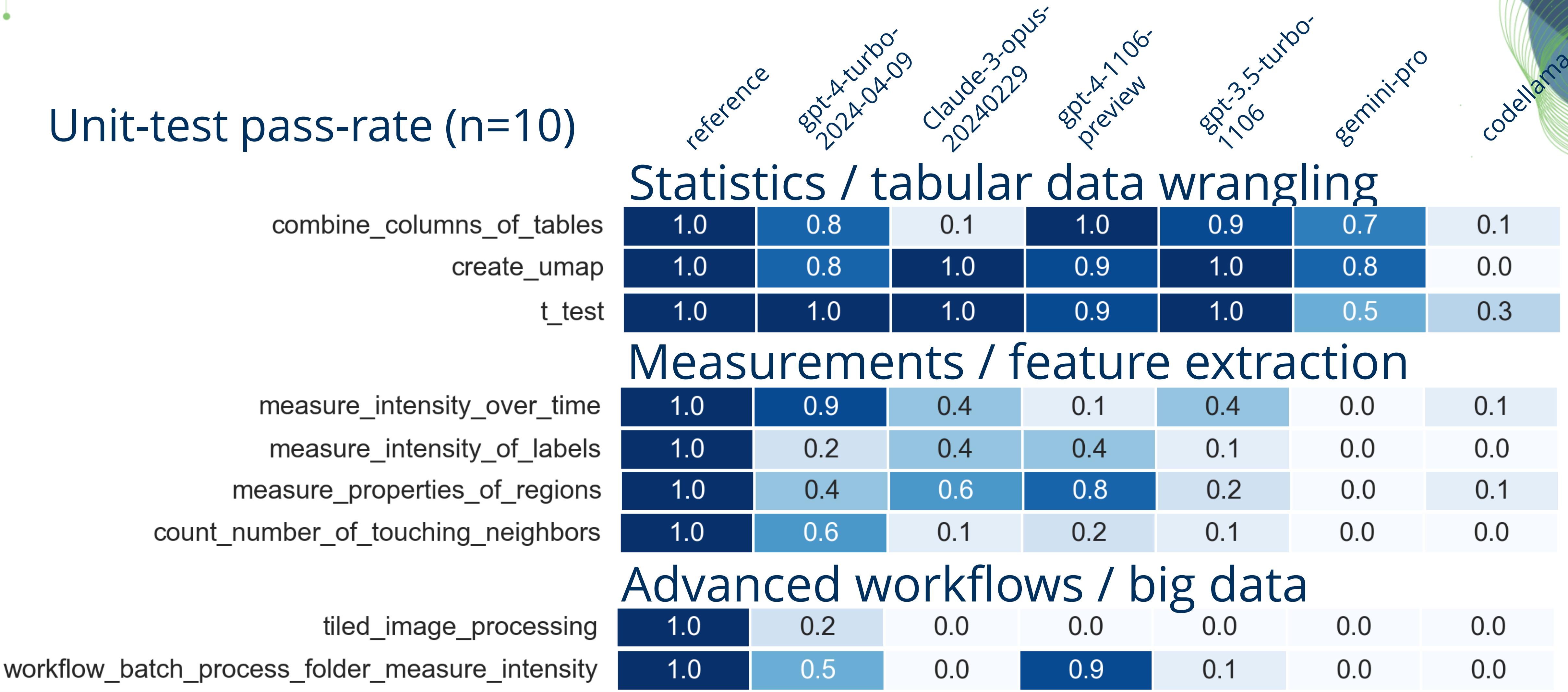
300
398
368
378
363

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

Benchmarking LLMs for Bio-image Analysis



Unit-test pass-rate (n=10)

Statistics / tabular data wrangling

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
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

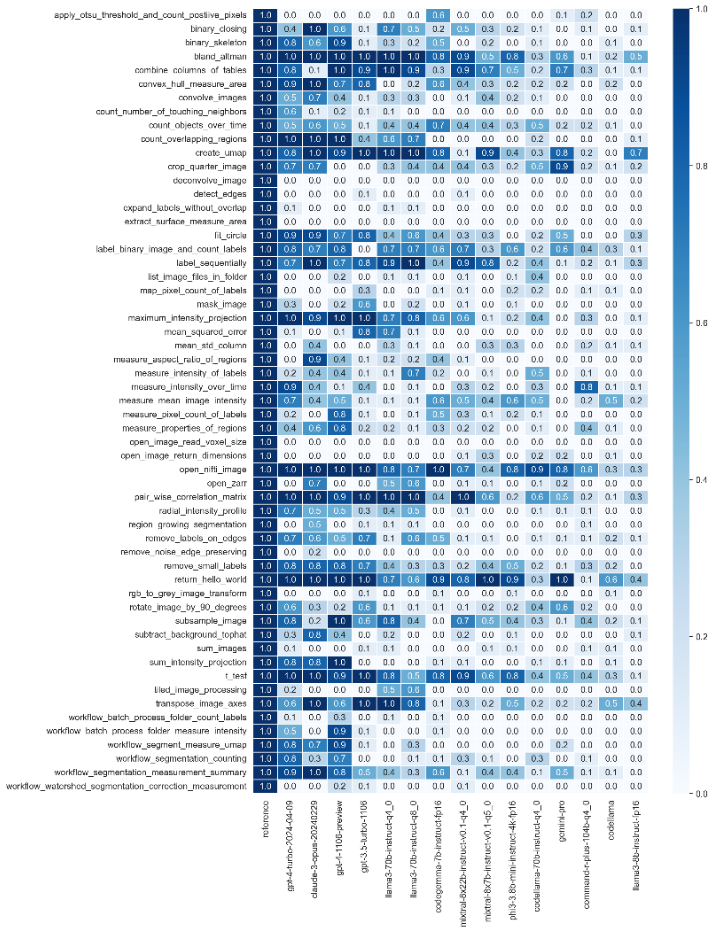
	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
measure_intensity_over_time	1.0	0.9	0.4	0.1	0.4	0.0	0.1
measure_intensity_of_labels	1.0	0.2	0.4	0.4	0.1	0.0	0.0
measure_properties_of_regions	1.0	0.4	0.6	0.8	0.2	0.0	0.1
count_number_of_touching_neighbors	1.0	0.6	0.1	0.2	0.1	0.0	0.0

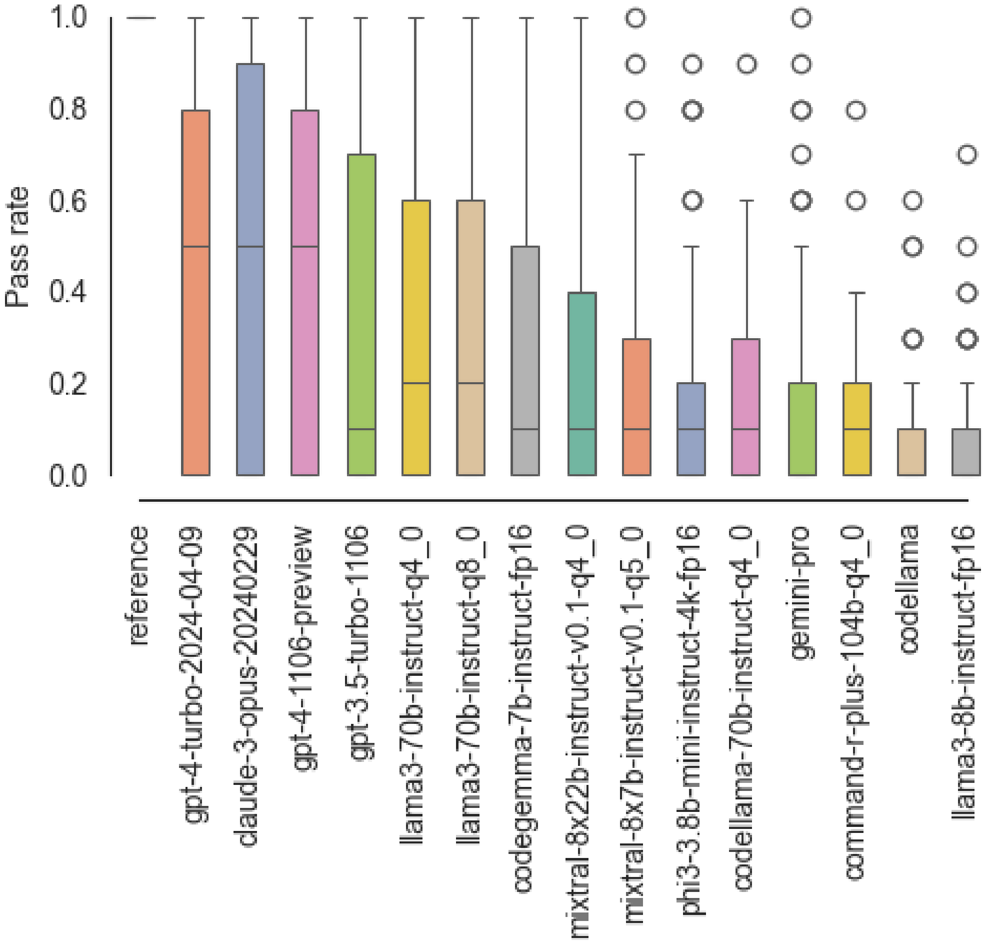
Advanced workflows / big data

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
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

Benchmarking LLMs for Bio-image Analysis

Summary: 57 use-cases (yet), 15 LLMs (yet), n=10

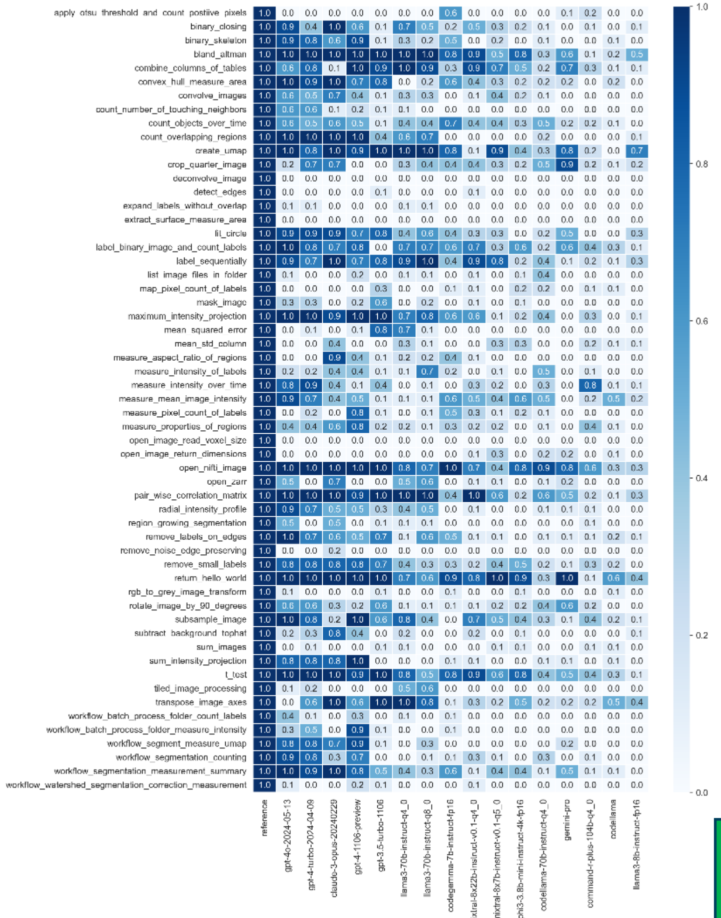




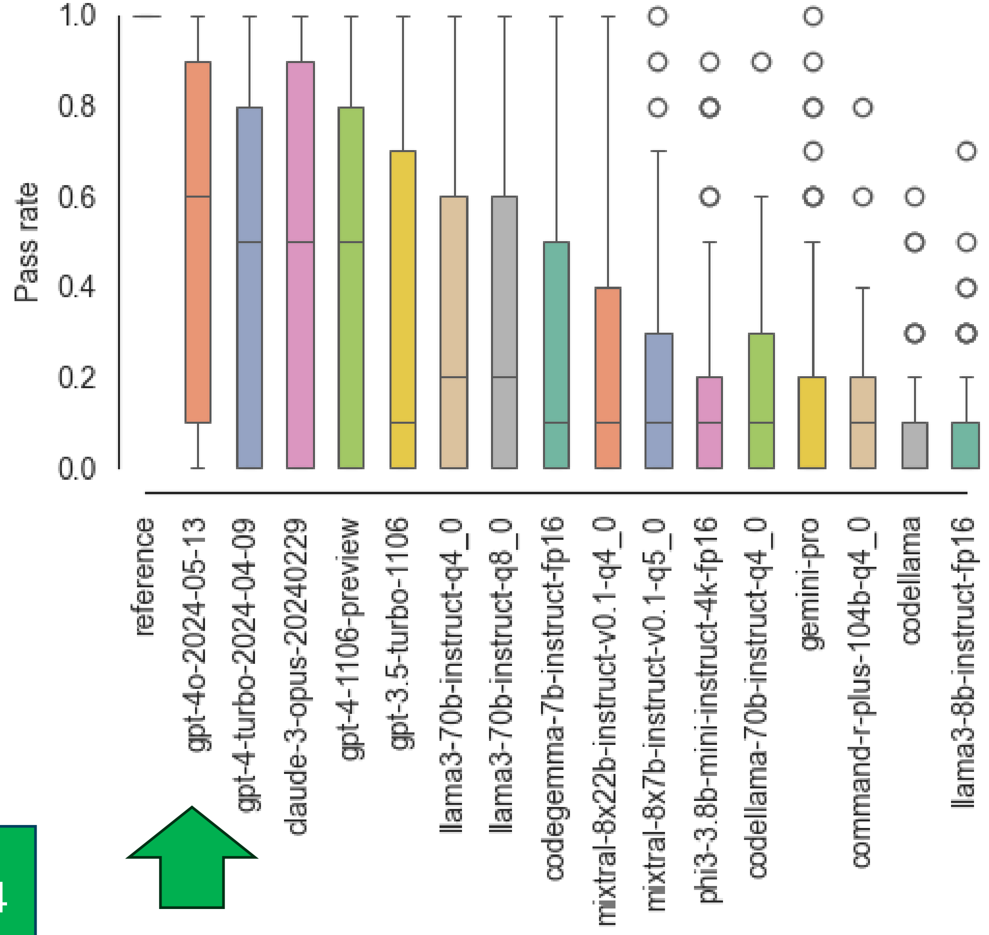
https://www.biorxiv.org/content/10.1101/2024.04.19.590278v1
https://github.com/haesleinhuepf/human-eval-bia

Benchmarking LLMs for Bio-image Analysis

Summary: 57 use-cases (yet), 16 LLMs (yet), n=10



	reference	gpt-4o-2024-05-13	gpt-4-turbo-2024-04-09	claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	llama3-70b-instruct-q4_0	llama3-70b-instruct-q8_0	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	command-r-plus-104b-q4_0	codellama	llama3-8b-instruct-fp16
apply otsu threshold and count postiive pixels	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.0
binary_closing	1.0	0.9	0.4	1.0	0.6	0.1	0.7	0.5	0.2	0.5	0.3	0.2	0.1	0.0	0.1	0.0	0.1
binary_skeleton	1.0	0.9	0.8	0.6	0.9	0.1	0.3	0.2	0.5	0.0	0.2	0.0	0.1	0.0	0.0	0.1	0.0
bland_altman	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.9	0.5	0.8	0.3	0.6	0.1	0.2	0.5
combine_columns_of_tables	1.0	0.6	0.8	0.1	1.0	0.9	1.0	0.9	0.3	0.9	0.7	0.5	0.2	0.7	0.3	0.1	0.1
convex_hull_measure_area	1.0	1.0	0.9	1.0	0.7	0.8	0.0	0.2	0.6	0.4	0.3	0.2	0.2	0.2	0.0	0.2	0.0
convolve_images	1.0	0.6	0.5	0.7	0.4	0.1	0.3	0.3	0.0	0.1	0.4	0.2	0.1	0.0	0.0	0.0	0.0
count_number_of_touching_neighbors	1.0	0.6	0.6	0.1	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
count_objects_over_time	1.0	0.6	0.5	0.6	0.5	0.1	0.4	0.4	0.7	0.4	0.4	0.3	0.5	0.2	0.2	0.1	0.0
count_overlapping_regions	1.0	1.0	1.0	1.0	1.0	0.4	0.6	0.7	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.1
create_umap	1.0	1.0	0.8	1.0	0.9	1.0	1.0	1.0	0.8	0.1	0.9	0.4	0.3	0.8	0.2	0.0	0.7
crop_quarter_image	1.0	0.2	0.7	0.7	0.0	0.0	0.3	0.4	0.4	0.4	0.3	0.2	0.5	0.9	0.2	0.1	0.2
deconvolve_image	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
detect_edges	1.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
expand_labels_without_overlap	1.0	0.1	0.1	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
extract_surface_measure_area	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
fit_circle	1.0	0.9	0.9	0.9	0.7	0.8	0.4	0.6	0.4	0.3	0.3	0.0	0.2	0.5	0.0	0.0	0.3
label_binary_image_and_count_labels	1.0	1.0	0.8	0.7	0.8	0.0	0.7	0.7	0.6	0.7	0.3	0.6	0.2	0.6	0.4	0.3	0.1
label_sequentially	1.0	0.9	0.7	1.0	0.7	0.8	0.9	1.0	0.4	0.9	0.8	0.2	0.4	0.1	0.2	0.1	0.3
list image files in folder	1.0	0.1	0.0	0.0	0.2	0.0	0.1	0.1	0.0	0.1	0.0	0.1	0.4	0.0	0.0	0.0	0.0
map_pixel_count_of_labels	1.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.1	0.1	0.0	0.2	0.2	0.0	0.1	0.1	0.0
mask_image	1.0	0.3	0.3	0.0	0.2	0.6	0.0	0.2	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0
maximum_intensity_projection	1.0	1.0	1.0	0.9	1.0	1.0	0.7	0.8	0.6	0.6	0.1	0.2	0.4	0.0	0.3	0.0	0.1
mean squared error	1.0	0.0	0.1	0.0	0.1	0.8	0.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
mean_std_column	1.0	0.0	0.0	0.4	0.0	0.0	0.3	0.1	0.0	0.0	0.3	0.3	0.0	0.0	0.2	0.1	0.1
measure_aspect_ratio_of_regions	1.0	0.0	0.0	0.9	0.4	0.1	0.2	0.2	0.4	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
measure_intensity_of_labels	1.0	0.2	0.2	0.4	0.4	0.1	0.1	0.7	0.2	0.0	0.1	0.0	0.5	0.0	0.1	0.0	0.0
measure intensity over time	1.0	0.8	0.9	0.4	0.1	0.4	0.0	0.1	0.0	0.3	0.2	0.0	0.3	0.0	0.8	0.1	0.1
measure_mean_image_intensity	1.0	0.9	0.7	0.4	0.5	0.1	0.1	0.1	0.6	0.5	0.4	0.6	0.5	0.0	0.2	0.5	0.2
measure_pixel_count_of_labels	1.0	0.0	0.2	0.0	0.8	0.1	0.0	0.1	0.5	0.3	0.1	0.2	0.1	0.0	0.0	0.0	0.0
measure_properties_of_regions	1.0	0.4	0.4	0.6	0.8	0.2	0.2	0.1	0.3	0.2	0.2	0.0	0.1	0.0	0.4	0.1	0.0
open_image_read_voxel_size	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
open_image_return_dimensions	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.3	0.0	0.2	0.2	0.0	0.1	0.0
open_nifti_image	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.7	1.0	0.7	0.4	0.8	0.9	0.8	0.6	0.3	0.3
open_zarr	1.0	0.5	0.0	0.7	0.0	0.0	0.5	0.6	0.0	0.1	0.1	0.0	0.1	0.2	0.0	0.0	0.0
pair_wise_correlation_matrix	1.0	1.0	1.0	1.0	0.9	1.0	1.0	1.0	0.4	1.0	0.6	0.2	0.6	0.5	0.2	0.1	0.3
radial_intensity_profile	1.0	0.9	0.7	0.5	0.5	0.3	0.4	0.5	0.0	0.1	0.0	0.0	0.1	0.1	0.1	0.0	0.0
region_growing_segmentation	1.0	0.5	0.0	0.5	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0
remove_labels_on_edges	1.0	1.0	0.7	0.6	0.5	0.7	0.1	0.6	0.5	0.1	0.1	0.0	0.1	0.1	0.1	0.2	0.1
remove_noise_edge_preserving	1.0	0.0	0.0	0.2	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
remove_small_labels	1.0	0.8	0.8	0.8	0.8	0.7	0.4	0.3	0.3	0.2	0.4	0.5	0.2	0.1	0.3	0.2	0.0
return_hello_world	1.0	1.0	1.0	1.0	1.0	1.0	0.7	0.6	0.9	0.8	1.0	0.9	0.3	1.0	0.1	0.6	0.4
rgb_to_grey_image_transform	1.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.0
rotate_image_by_90_degrees	1.0	0.6	0.6	0.3	0.2	0.6	0.1	0.1	0.1	0.1	0.2	0.2	0.4	0.6	0.2	0.0	0.0
subsample_image	1.0	1.0	0.8	0.2	1.0	0.6	0.8	0.4	0.0	0.7	0.5	0.4	0.3	0.1	0.4	0.2	0.1
subtract background tophat	1.0	0.2	0.3	0.8	0.4	0.0	0.2	0.0	0.0	0.2	0.0	0.1	0.0	0.0	0.0	0.0	0.1
sum_images	1.0	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.1	0.1	0.0
sum_intensity_projection	1.0	0.8	0.8	0.8	1.0	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.1	0.1	0.0	0.1	0.0
t_test	1.0	1.0	1.0	1.0	0.9	1.0	0.8	0.5	0.8	0.9	0.6	0.8	0.4	0.5	0.4	0.3	0.1
tiled_image_processing	1.0	0.1	0.2	0.0	0.0	0.0	0.5	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
transpose_image_axes	1.0	0.0	0.6	1.0	0.6	1.0	1.0	0.8	0.1	0.3	0.2	0.5	0.2	0.2	0.2	0.5	0.4
workflow_batch_process_folder_count_labels	1.0	0.4	0.1	0.0	0.3	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
workflow_batch_process_folder_measure_intensity	1.0	0.3	0.5	0.0	0.9	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
workflow_segment_measure_umap	1.0	0.8	0.8	0.7	0.9	0.1	0.0	0.3	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0
workflow_segmentation_counting	1.0	0.9	0.8	0.3	0.7	0.0	0.0	0.1	0.1	0.3	0.1	0.0	0.3	0.0	0.1	0.0	0.0
workflow_segmentation_measurement_summary	1.0	1.0	0.9	1.0	0.8	0.5	0.4	0.3	0.6	0.1	0.4	0.4	0.1	0.5	0.1	0.1	0.0
workflow_watershed_segmentation_correction_measurement	1.0	0.1	0.0	0.0	0.2	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0

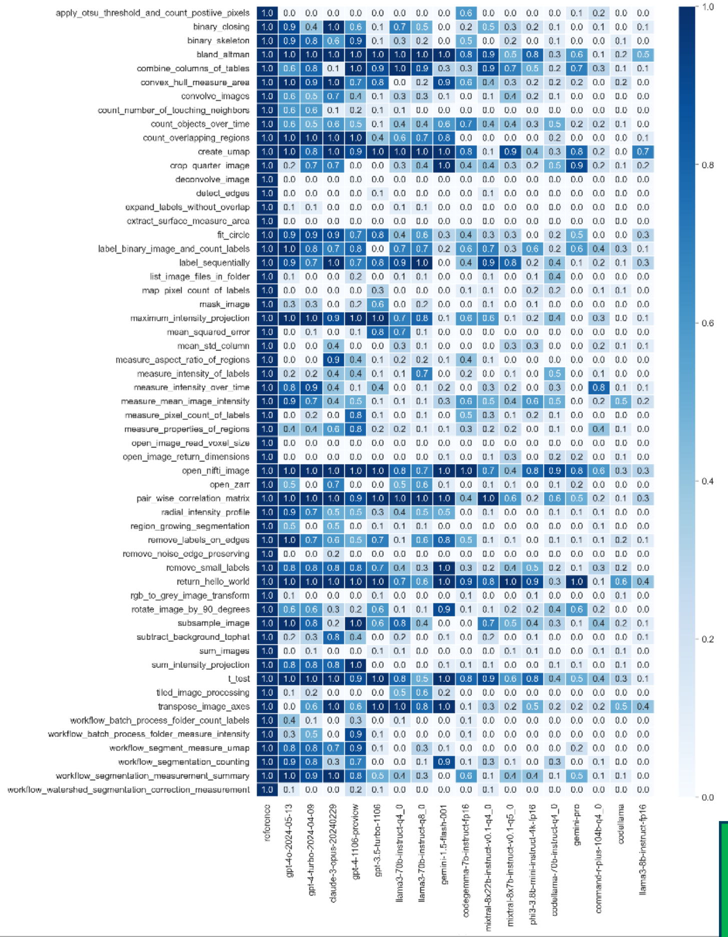


New: gpt4 omni

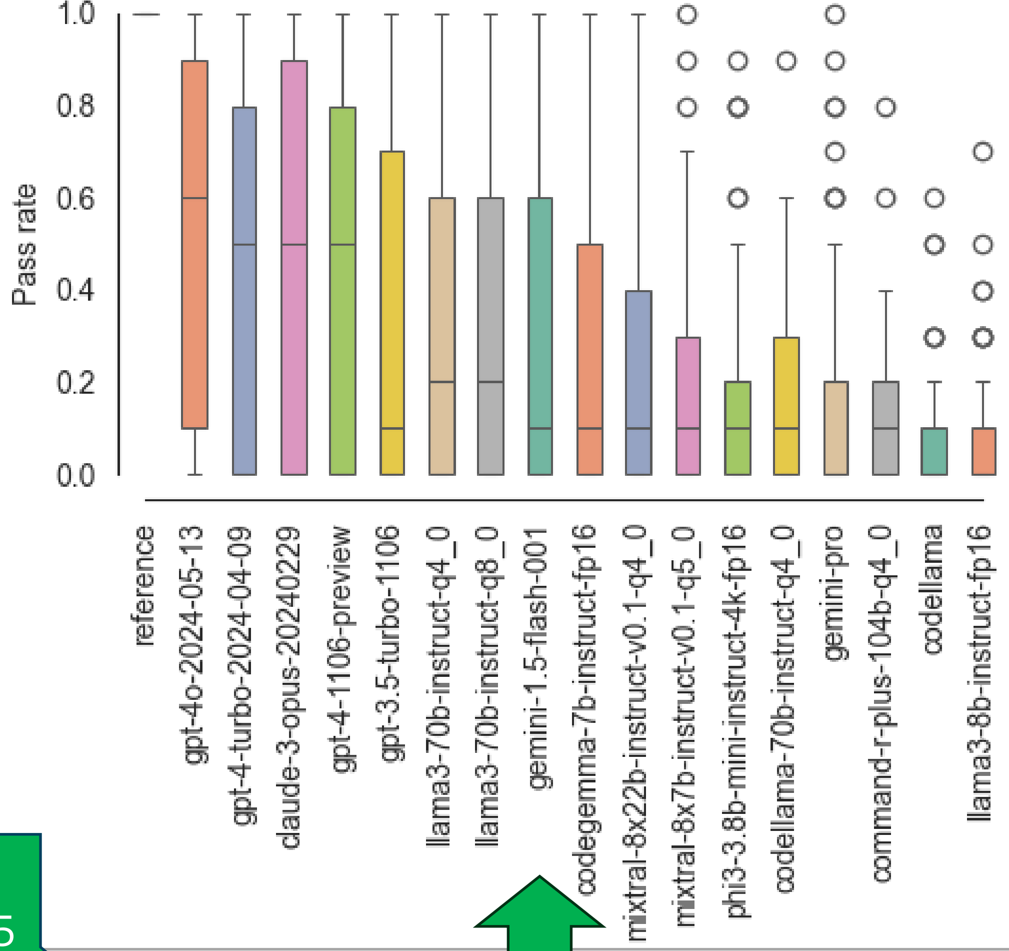
https://github.com/haesleinhuepf/human-eval-bia/pull/66

Benchmarking LLMs for Bio-image Analysis

Summary: 57 use-cases (yet), 17 LLMs (yet), n=10



	reference	gpt-4o-2024-05-13	gpt-4-turbo-2024-04-09	claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	llama3-70b-instruct-q4_0	llama3-70b-instruct-q8_0	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	command-r-plus-104b-q4_0	codellama	llama3-8b-instruct-fp16
apply_otsu_threshold_and_count_postiive_pixels	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.0
binary_closing	1.0	0.9	0.4	1.0	0.6	0.1	0.7	0.5	0.0	0.2	0.5	0.3	0.2	0.1	0.0	0.1	0.0	0.1
binary_skeleton	1.0	0.9	0.8	0.6	0.9	0.1	0.3	0.2	0.0	0.5	0.0	0.2	0.0	0.1	0.0	0.0	0.1	0.0
bland_altman	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.9	0.5	0.8	0.3	0.6	0.1	0.2	0.5
combine_columns_of_tables	1.0	0.6	0.8	0.1	1.0	0.9	1.0	0.9	0.3	0.3	0.9	0.7	0.5	0.2	0.7	0.3	0.1	0.1
convex_hull_measure_area	1.0	1.0	0.9	1.0	0.7	0.8	0.0	0.2	0.9	0.6	0.4	0.3	0.2	0.2	0.2	0.0	0.2	0.0
convolve_images	1.0	0.6	0.5	0.7	0.4	0.1	0.3	0.3	0.1	0.0	0.1	0.4	0.2	0.1	0.0	0.0	0.0	0.0
count_number_of_touching_neighbors	1.0	0.6	0.6	0.1	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
count_objects_over_time	1.0	0.6	0.5	0.6	0.5	0.1	0.4	0.4	0.6	0.7	0.4	0.4	0.3	0.5	0.2	0.2	0.1	0.0
count_overlapping_regions	1.0	1.0	1.0	1.0	1.0	0.4	0.6	0.7	0.8	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.1
create_umap	1.0	1.0	0.8	1.0	0.9	1.0	1.0	1.0	1.0	0.8	0.1	0.9	0.4	0.3	0.8	0.2	0.0	0.7
crop_quarter_image	1.0	0.2	0.7	0.7	0.0	0.0	0.3	0.4	1.0	0.4	0.4	0.3	0.2	0.5	0.9	0.2	0.1	0.2
deconvolve_image	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
detect_edges	1.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
expand_labels_without_overlap	1.0	0.1	0.1	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
extract_surface_measure_area	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
fit_circle	1.0	0.9	0.9	0.9	0.7	0.8	0.4	0.6	0.3	0.4	0.3	0.3	0.0	0.2	0.5	0.0	0.0	0.3
label_binary_image_and_count_labels	1.0	1.0	0.8	0.7	0.8	0.0	0.7	0.7	0.2	0.6	0.7	0.3	0.6	0.2	0.6	0.4	0.3	0.1
label_sequentially	1.0	0.9	0.7	1.0	0.7	0.8	0.9	1.0	0.0	0.4	0.9	0.8	0.2	0.4	0.1	0.2	0.1	0.3
list_image_files_in_folder	1.0	0.1	0.0	0.0	0.2	0.0	0.1	0.1	0.0	0.0	0.1	0.0	0.1	0.4	0.0	0.0	0.0	0.0
map_pixel_count_of_labels	1.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.1	0.1	0.0	0.2	0.2	0.0	0.1	0.1	0.0
mask_image	1.0	0.3	0.3	0.0	0.2	0.6	0.0	0.2	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0
maximum_intensity_projection	1.0	1.0	1.0	0.9	1.0	1.0	0.7	0.8	0.1	0.6	0.6	0.1	0.2	0.4	0.0	0.3	0.0	0.1
mean_squared_error	1.0	0.0	0.1	0.0	0.1	0.8	0.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
mean_std_column	1.0	0.0	0.0	0.4	0.0	0.0	0.3	0.1	0.0	0.0	0.0	0.3	0.3	0.0	0.0	0.2	0.1	0.1
measure_aspect_ratio_of_regions	1.0	0.0	0.0	0.9	0.4	0.1	0.2	0.2	0.1	0.4	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
measure_intensity_of_labels	1.0	0.2	0.2	0.4	0.4	0.1	0.1	0.7	0.0	0.2	0.0	0.1	0.0	0.5	0.0	0.1	0.0	0.0
measure_intensity_over_time	1.0	0.8	0.9	0.4	0.1	0.4	0.0	0.1	0.2	0.0	0.3	0.2	0.0	0.3	0.0	0.8	0.1	0.1
measure_mean_image_intensity	1.0	0.9	0.7	0.4	0.5	0.1	0.1	0.1	0.3	0.6	0.5	0.4	0.6	0.5	0.0	0.2	0.5	0.2
measure_pixel_count_of_labels	1.0	0.0	0.2	0.0	0.8	0.1	0.0	0.1	0.0	0.5	0.3	0.1	0.2	0.1	0.0	0.0	0.0	0.0
measure_properties_of_regions	1.0	0.4	0.4	0.6	0.8	0.2	0.2	0.1	0.1	0.3	0.2	0.2	0.0	0.1	0.0	0.4	0.1	0.0
open_image_read_voxel_size	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
open_image_return_dimensions	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.3	0.0	0.2	0.2	0.0	0.1	0.0
open_nifti_image	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.7	1.0	1.0	0.7	0.4	0.8	0.9	0.8	0.6	0.3	0.3
open_zarr	1.0	0.5	0.0	0.7	0.0	0.0	0.5	0.6	0.1	0.0	0.1	0.1	0.0	0.1	0.2	0.0	0.0	0.0
pair_wise_correlation_matrix	1.0	1.0	1.0	1.0	0.9	1.0	1.0	1.0	1.0	0.4	1.0	0.6	0.2	0.6	0.5	0.2	0.1	0.3
radial_intensity_profile	1.0	0.9	0.7	0.5	0.5	0.3	0.4	0.5	0.5	0.0	0.1	0.0	0.0	0.1	0.1	0.1	0.0	0.0
region_growing_segmentation	1.0	0.5	0.0	0.5	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0
remove_labels_on_edges	1.0	1.0	0.7	0.6	0.5	0.7	0.1	0.6	0.8	0.5	0.1	0.1	0.0	0.1	0.1	0.1	0.2	0.1
remove_noise_edge_preserving	1.0	0.0	0.0	0.2	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	0.0
remove_small_labels	1.0	0.8	0.8	0.8	0.8	0.7	0.4	0.3	1.0	0.3	0.2	0.4	0.5	0.2	0.1	0.3	0.2	0.0
return_hello_world	1.0	1.0	1.0	1.0	1.0	1.0	0.7	0.6	1.0	0.9	0.8	1.0	0.9	0.3	1.0	0.1	0.6	0.4
rgb_to_grey_image_transform	1.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.0
rotate_image_by_90_degrees	1.0	0.6	0.6	0.3	0.2	0.6	0.1	0.1	0.9	0.1	0.1	0.2	0.2	0.4	0.6	0.2	0.0	0.0
subsample_image	1.0	1.0	0.8	0.2	1.0	0.6	0.8	0.4	0.0	0.0	0.7	0.5	0.4	0.3	0.1	0.4	0.2	0.1
subtract_background_tophat	1.0	0.2	0.3	0.8	0.4	0.0	0.2	0.0	0.1	0.0	0.2	0.0	0.1	0.0	0.0	0.0	0.0	0.1
sum_images	1.0	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.1	0.1	0.0
sum_intensity_projection	1.0	0.8	0.8	0.8	1.0	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.0	0.1	0.1	0.0	0.1	0.0
t_test	1.0	1.0	1.0	1.0	0.9	1.0	0.8	0.5	1.0	0.8	0.9	0.6	0.8	0.4	0.5	0.4	0.3	0.1
tiled_image_processing	1.0	0.1	0.2	0.0	0.0	0.0	0.5	0.6	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
transpose_image_axes	1.0	0.0	0.6	1.0	0.6	1.0	1.0	0.8	1.0	0.1	0.3	0.2	0.5	0.2	0.2	0.2	0.5	0.4
workflow_batch_process_folder_count_labels	1.0	0.4	0.1	0.0	0.3	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
workflow_batch_process_folder_measure_intensity	1.0	0.3	0.5	0.0	0.9	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
workflow_segment_measure_umap	1.0	0.8	0.8	0.7	0.9	0.1	0.0	0.3	0.1	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0
workflow_segmentation_counting	1.0	0.9	0.8	0.3	0.7	0.0	0.0	0.1	0.9	0.1	0.3	0.1	0.0	0.3	0.0	0.1	0.0	0.0
workflow_segmentation_measurement_summary	1.0	1.0	0.9	1.0	0.8	0.5	0.4	0.3	0.0	0.6	0.1	0.4	0.4	0.1	0.5	0.1	0.1	0.0
workflow_watershed_segmentation_correction_measurement	1.0	0.1	0.0	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0

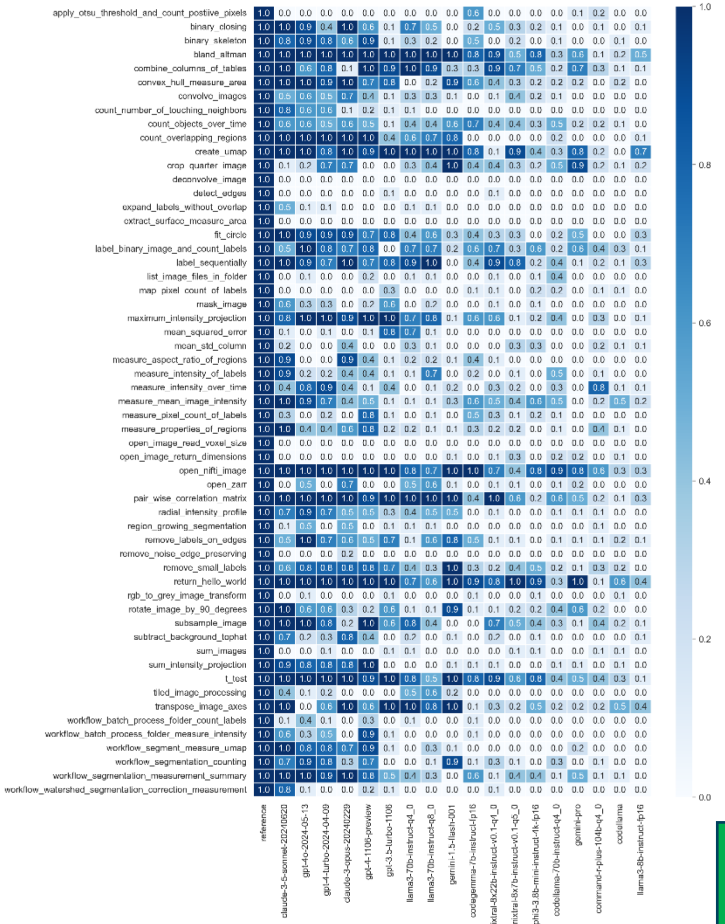


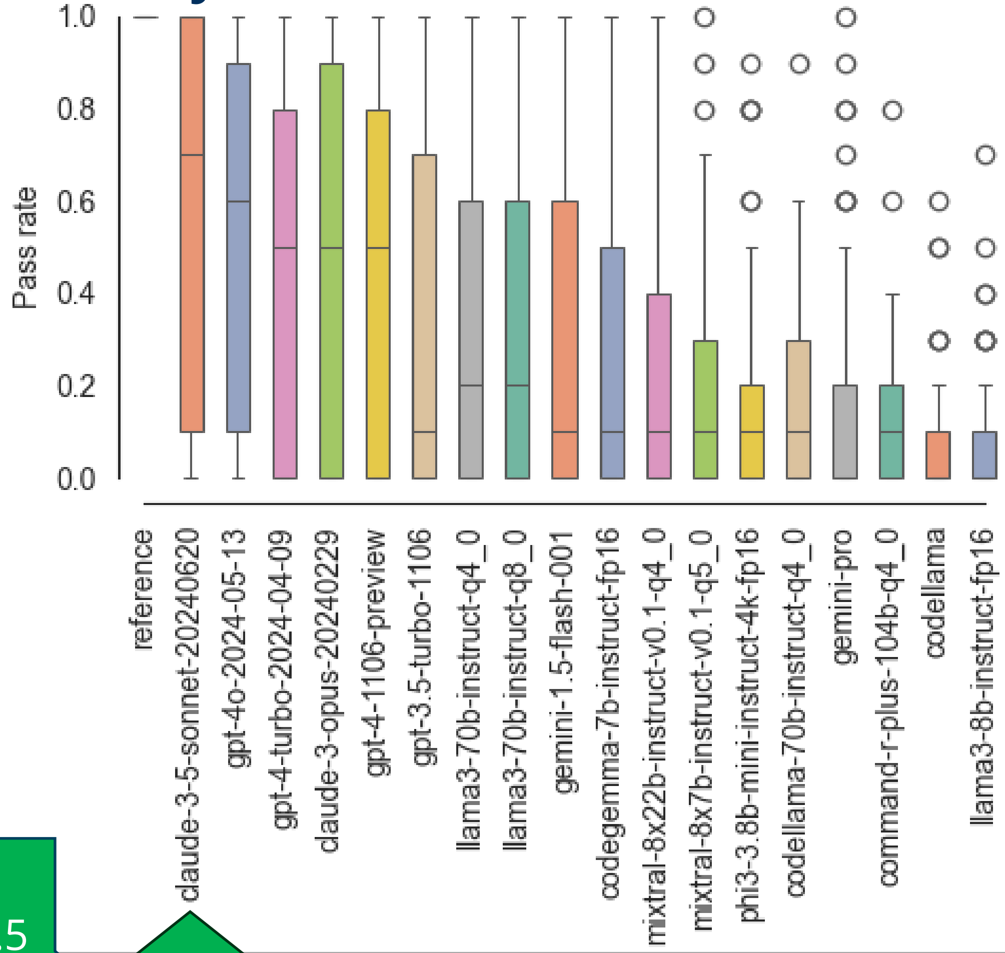
New:
gemini 1.5
flash

https://github.com/haesleinhuepf/human-eval-bia/pull/67

Benchmarking LLMs for Bio-image Analysis

Summary: 57 use-cases (yet), 18 LLMs (yet), n=10





New:
claude 3.5
sonnet!

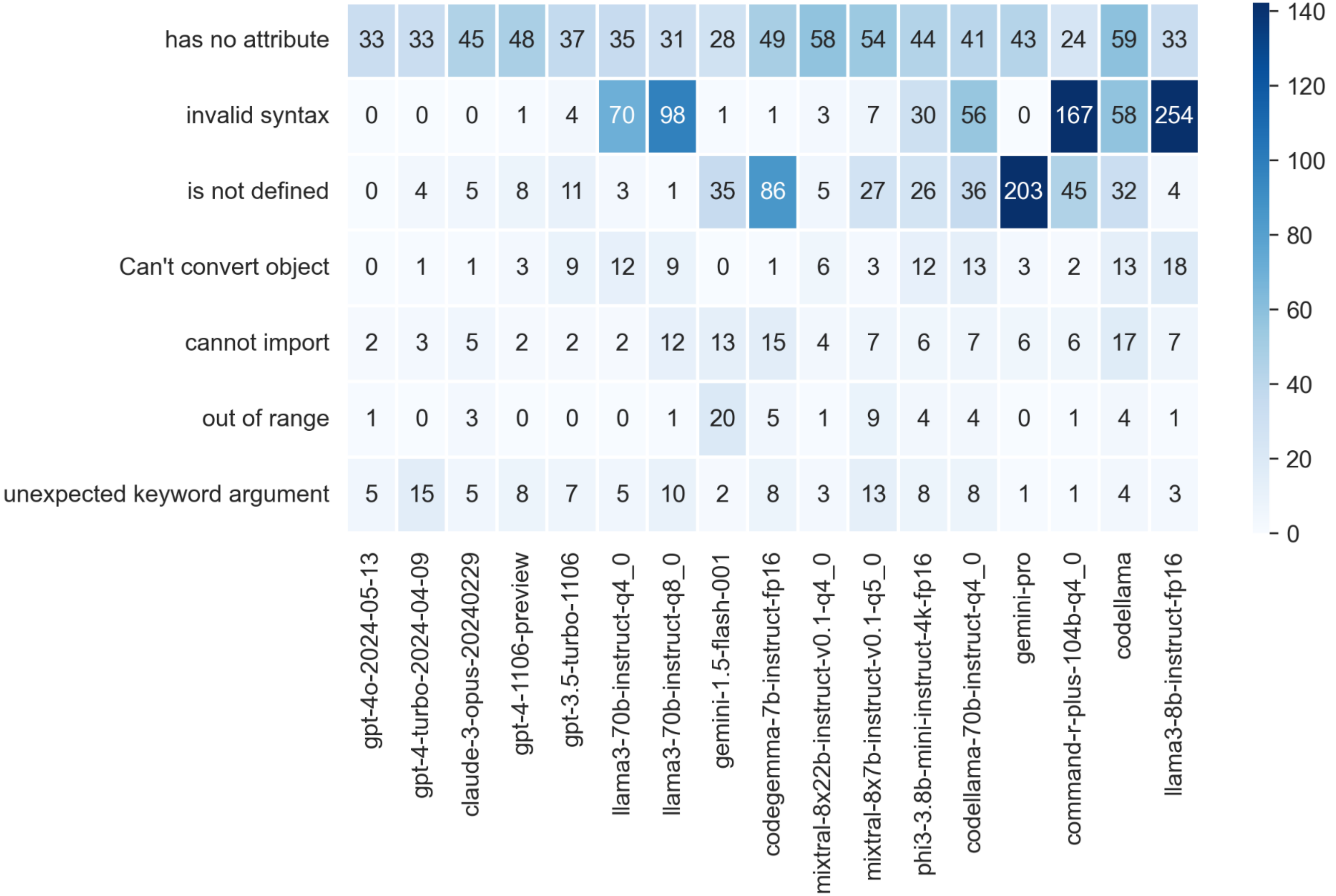
https://github.com/haesleinhuepf/human-eval-bia/pull/69

Benchmarking LLMs for Bio-image Analysis

Common error messages (n=570)

Halucinating API?

Forgot import statements?

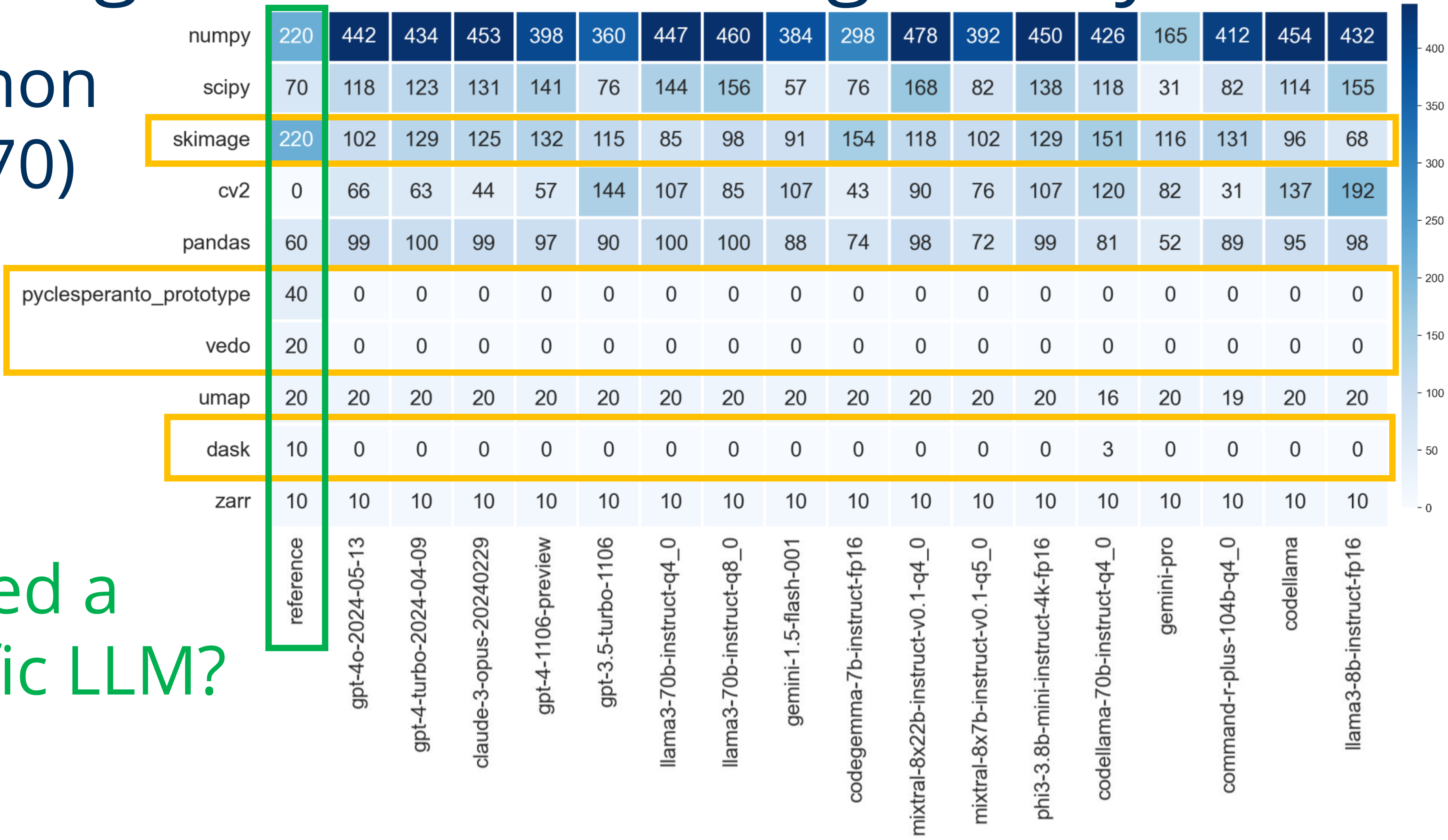


	gpt-4o-2024-05-13	gpt-4-turbo-2024-04-09	claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	llama3-70b-instruct-q4_0	llama3-70b-instruct-q8_0	gemini-1.5-flash-001	codegemma-7b-instruct-fp16	mixtral-8x22b-instruct-v0.1-q4_0	mixtral-8x7b-instruct-v0.1-q5_0	phi3-3.8b-mini-instruct-4k-fp16	codellama-70b-instruct-q4_0	gemini-pro	command-r-plus-104b-q4_0	codellama	llama3-8b-instruct-fp16
has no attribute	33	33	45	48	37	35	31	28	49	58	54	44	41	43	24	59	33
invalid syntax	0	0	0	1	4	70	98	1	1	3	7	30	56	0	167	58	254
is not defined	0	4	5	8	11	3	1	35	86	5	27	26	36	203	45	32	4
Can't convert object	0	1	1	3	9	12	9	0	1	6	3	12	13	3	2	13	18
cannot import	2	3	5	2	2	2	12	13	15	4	7	6	7	6	6	17	7
out of range	1	0	3	0	0	0	1	20	5	1	9	4	4	0	1	4	1
unexpected keyword argument	5	15	5	8	7	5	10	2	8	3	13	8	8	1	1	4	3

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

Benchmarking LLMs for Bio-image Analysis

Common Python libraries (n=570)



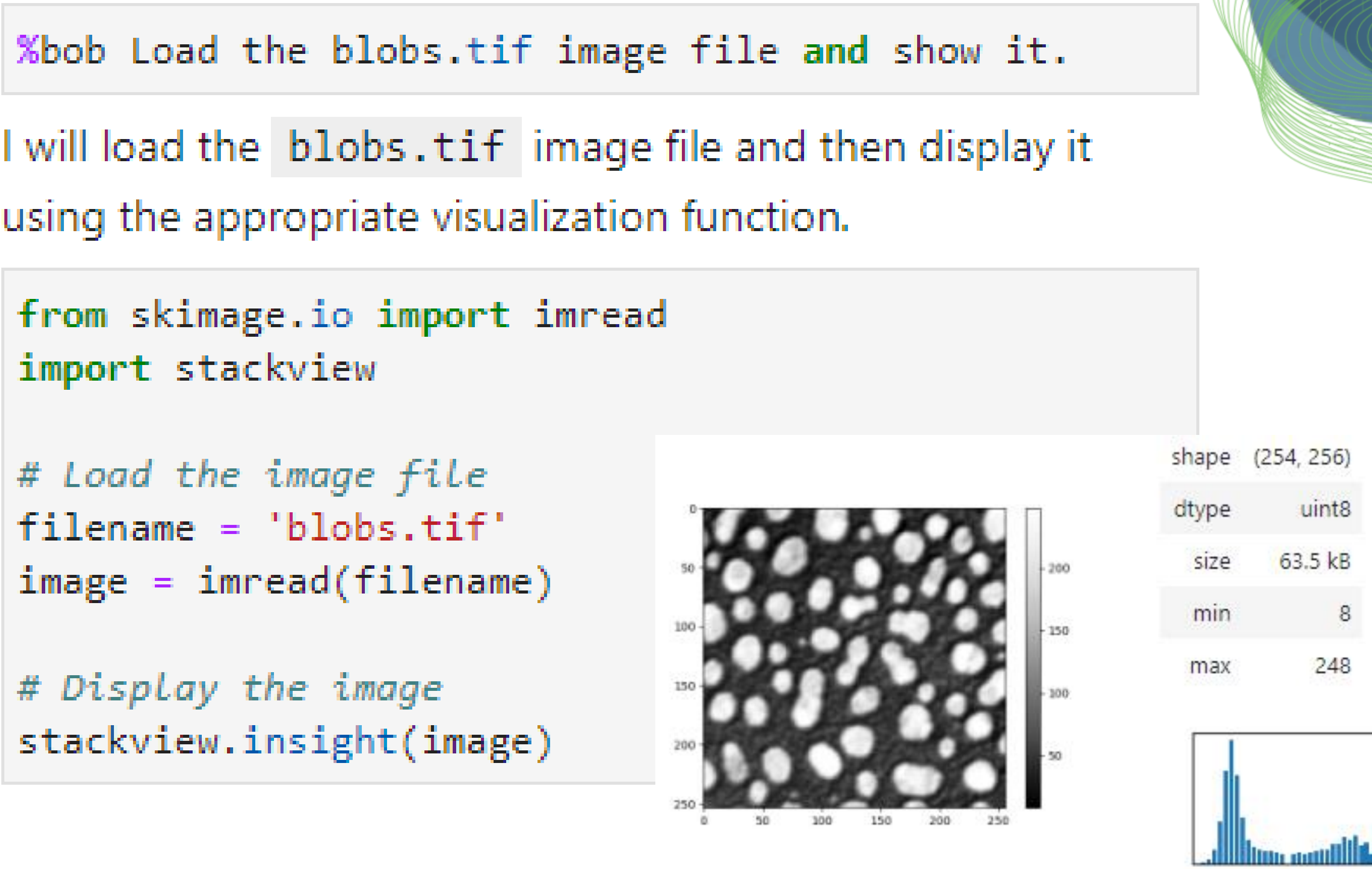
	reference	gpt-4o-2024-05-13	gpt-4-turbo-2024-04-09	claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	llama3-70b-instruct-q4_0	llama3-70b-instruct-q8_0	gemini-1.5-flash-001	codegemma-7b-instruct-fp16	mixtral-8x22b-instruct-v0.1-q4_0	mixtral-8x7b-instruct-v0.1-q5_0	phi3-3.8b-mini-instruct-4k-fp16	codellama-70b-instruct-q4_0	gemini-pro	command-r-plus-104b-q4_0	codellama	llama3-8b-instruct-fp16
numpy	220	442	434	453	398	360	447	460	384	298	478	392	450	426	165	412	454	432
scipy	70	118	123	131	141	76	144	156	57	76	168	82	138	118	31	82	114	155
skimage	220	102	129	125	132	115	85	98	91	154	118	102	129	151	116	131	96	68
cv2	0	66	63	44	57	144	107	85	107	43	90	76	107	120	82	31	137	192
pandas	60	99	100	99	97	90	100	100	88	74	98	72	99	81	52	89	95	98
pyclesperanto_prototype	40	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
umap	20	20	20	20	20	20	20	20	20	20	20	20	20	16	20	19	20	20
dask	10	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
zarr	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

Do we need a BiA-specific LLM?

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

Open source Bio-image Analysis assistant: BiA-Bob

- No need to upload your [image] data (-> privacy)
- Multi-agent / -platform capable
- Vision models
- Extensible through plugins
- Integrated in Jupyter



```
%bob Load the blobs.tif image file and show it.
```

I will load the `blobs.tif` image file and then display it using the appropriate visualization function.

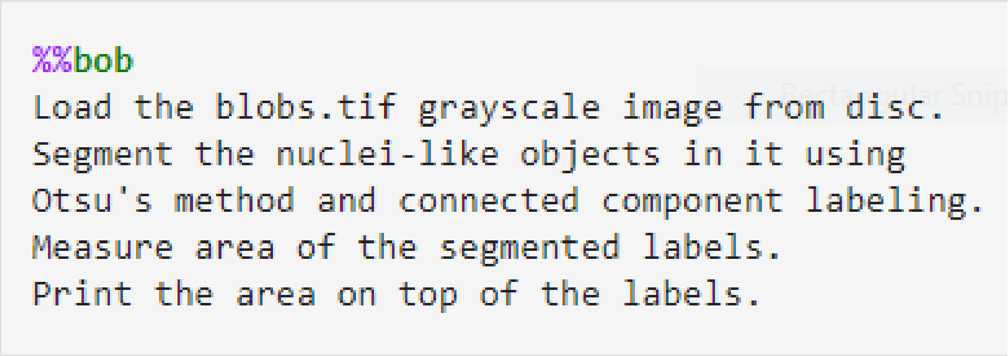
```
from skimage.io import imread
import stackview

# Load the image file
filename = 'blobs.tif'
image = imread(filename)

# Display the image
stackview.insight(image)
```

shape	(254, 256)
dtype	uint8
size	63.5 kB
min	8
max	248

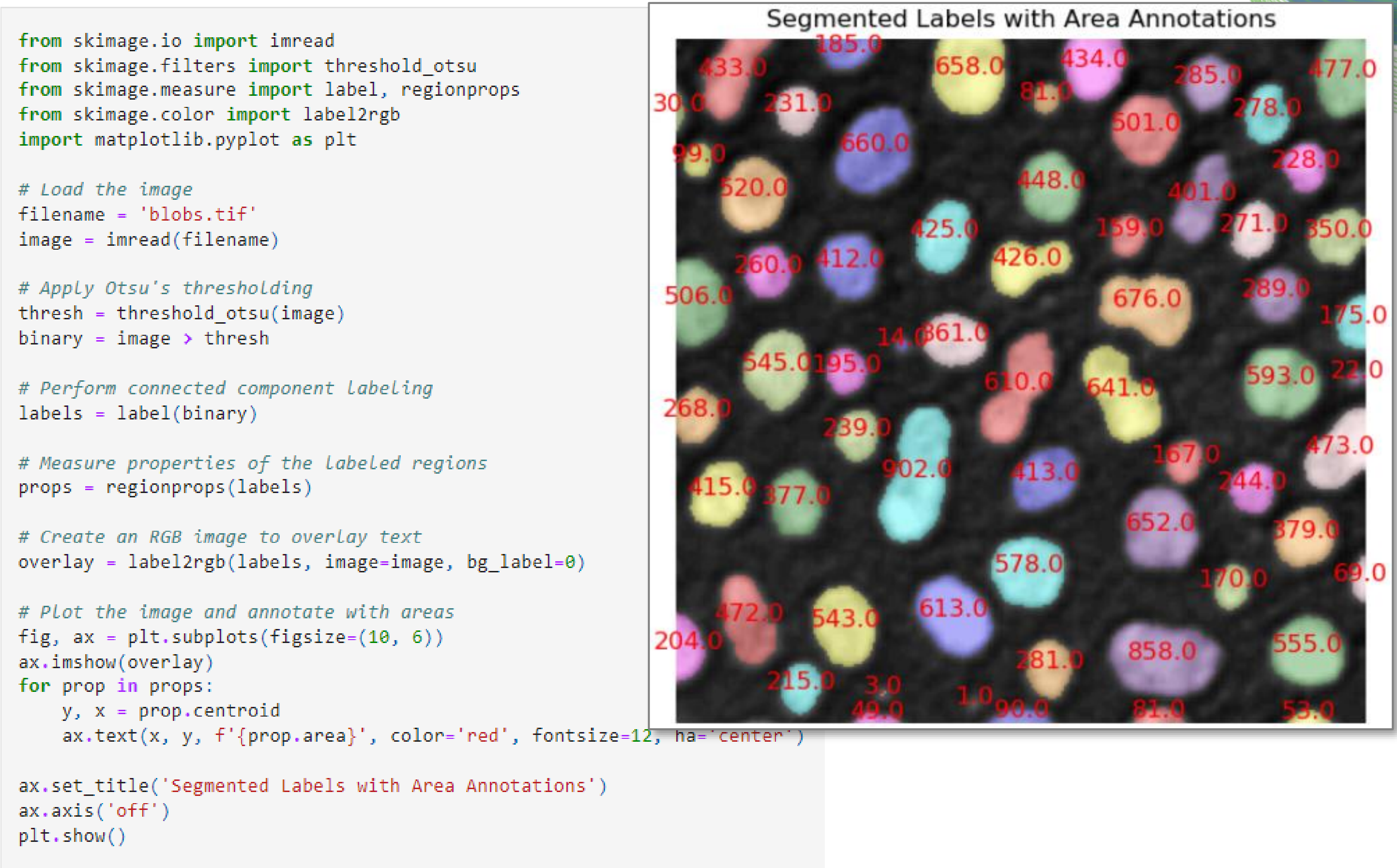
https://github.com/haesleinhuepf/bia-bob

Generating complex code blocks

Use the %%bob … syntax to ask for complex analysis tasks.



```
%%bob
Load the blobs.tif grayscale image from disc.
Segment the nuclei-like objects in it using
Otsu's method and connected component labeling.
Measure area of the segmented labels.
Print the area on top of the labels.
```



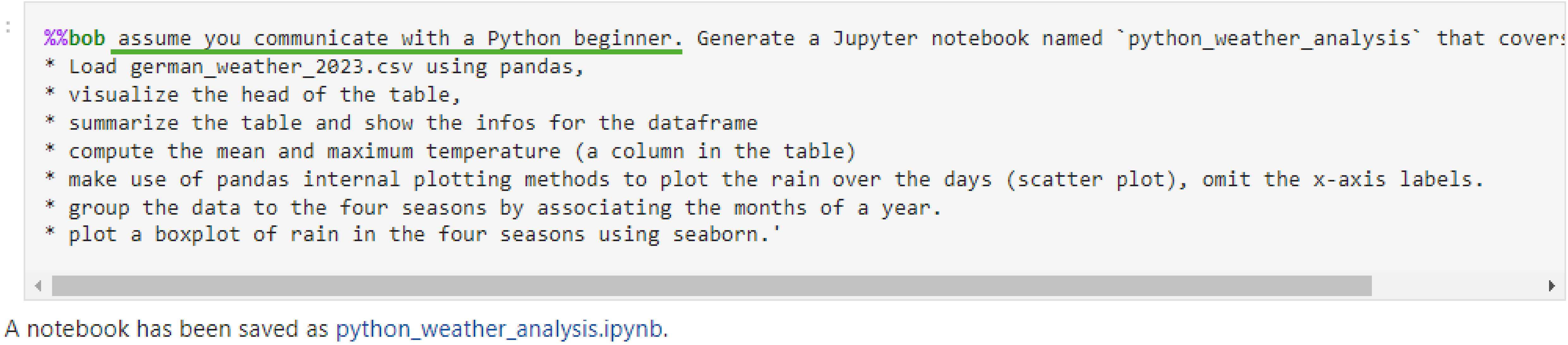
```python
from skimage.io import imread
from skimage.filters import threshold_otsu
from skimage.measure import label, regionprops
from skimage.color import label2rgb
import matplotlib.pyplot as plt

# Load the image
filename = 'blobs.tif'
image = imread(filename)

# Apply Otsu's thresholding
thresh = threshold_otsu(image)
binary = image > thresh

# Perform connected component labeling
labels = label(binary)

# Measure properties of the labeled regions
props = regionprops(labels)

# Create an RGB image to overlay text
overlay = label2rgb(labels, image=image, bg_label=0)

# Plot the image and annotate with areas
fig, ax = plt.subplots(figsize=(10, 6))
ax.imshow(overlay)
for prop in props:
    y, x = prop.centroid
    ax.text(x, y, f'{prop.area}', color='red', fontsize=12, ha='center')

ax.set_title('Segmented Labels with Area Annotations')
ax.axis('off')
plt.show()
```

Segmented Labels with Area Annotations

https://github.com/haesleinhuepf/bia-bob/blob/main/demo/basic_demo.ipynb

Generating notebooks

… also great for learning Python



```
%%bob assume you communicate with a Python beginner. Generate a Jupyter notebook named `python_weather_analysis` that cover
* Load german_weather_2023.csv using pandas,
* visualize the head of the table,
* summarize the table and show the infos for the dataframe
* compute the mean and maximum temperature (a column in the table)
* make use of pandas internal plotting methods to plot the rain over the days (scatter plot), omit the x-axis labels.
* group the data to the four seasons by associating the months of a year.
* plot a boxplot of rain in the four seasons using seaborn.'
```

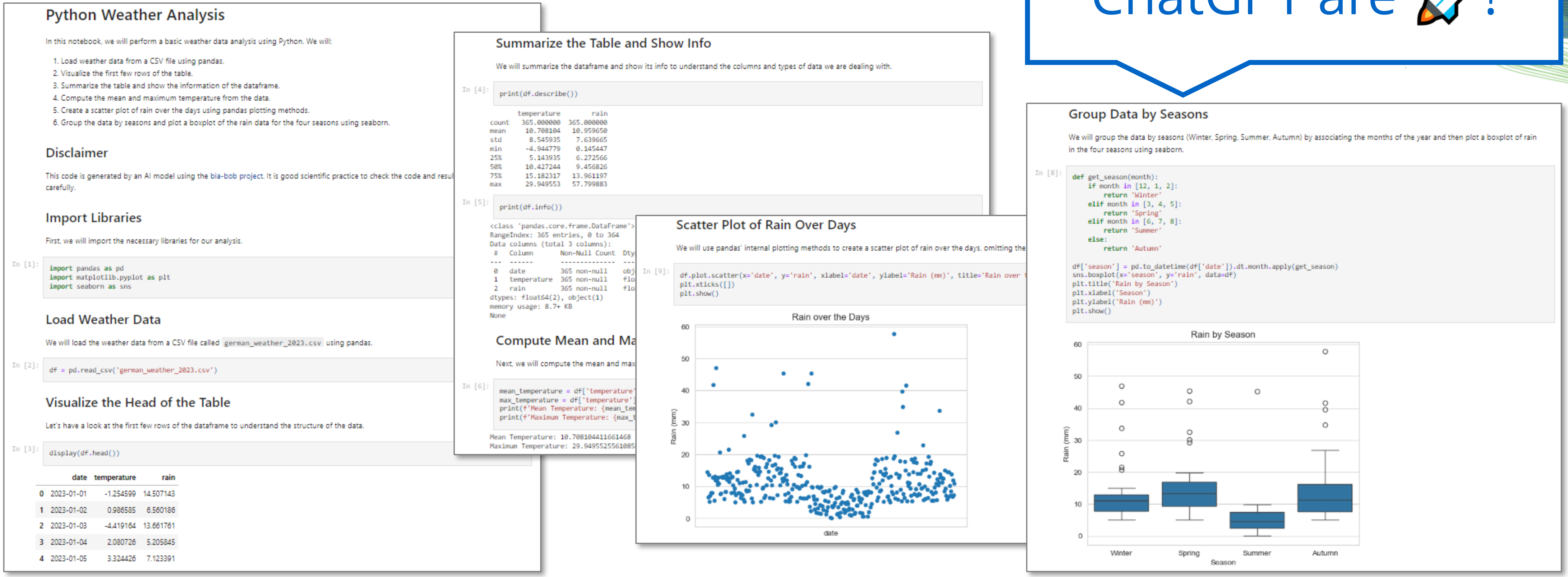
A notebook has been saved as python_weather_analysis.ipynb.

Cropped, licensed CC-BY 4.0 by Matthias Täschner & Robert Haase
https://github.com/ScaDS/Python-training-2024/blob/main/2_tables_plots/python_weather_analysis.ipynb

Generating notebooks

… also great for learning Python

In these kind of tasks, Python and ChatGPT are 🚀!



Python Weather Analysis

In this notebook, we will perform a basic weather data analysis using Python. We will:

1. Load weather data from a CSV file using pandas.
2. Visualize the first few rows of the table.
3. Summarize the table and show the information of the dataframe.
4. Compute the mean and maximum temperature from the data.
5. Create a scatter plot of rain over the days using pandas plotting methods.
6. Group the data by seasons and plot a boxplot of the rain data for the four seasons using seaborn.

Disclaimer

This code is generated by an AI model using the bia-bob project. It is good scientific practice to check the code and resul carefully.

Import Libraries

First, we will import the necessary libraries for our analysis.

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Weather Data

We will load the weather data from a CSV file called `german_weather_2023.csv` using pandas.

In [2]:

```
df = pd.read_csv('german_weather_2023.csv')
```

Visualize the Head of the Table

Let's have a look at the first few rows of the dataframe to understand the structure of the data.

In [3]:

```
display(df.head())
```

	date	temperature	rain
0	2023-01-01	-1.254599	14.507143
1	2023-01-02	0.986585	6.560186
2	2023-01-03	-4.419164	13.661761
3	2023-01-04	2.080726	5.205845
4	2023-01-05	3.324426	7.123391

Summarize the Table and Show Info

We will summarize the dataframe and show its info to understand the columns and types of data we are dealing with.

In [4]:

```
print(df.describe())
```

```
       temperature        rain
count   365.000000  365.000000
mean     10.708104   10.959650
std       8.545935    7.639665
min      -4.944779    0.145447
25%       5.143935    6.272566
50%      10.427244    9.456826
75%      15.182317   13.961197
max      29.949553   57.799883
```

In [5]:

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dty
---  ------       --------------  ---
 0   date         365 non-null    obj
 1   temperature  365 non-null    flo
 2   rain         365 non-null    flo
dtypes: float64(2), object(1)
memory usage: 8.7+ KB
None
```

Compute Mean and Ma

Next, we will compute the mean and max

In [6]:

```
mean_temperature = df['temperature'
max_temperature = df['temperature']
print(f'Mean Temperature: {mean_tem
print(f'Maximum Temperature: {max_t
```

```
Mean Temperature: 10.708104411661468
Maximum Temperature: 29.9495525561085
```

Scatter Plot of Rain Over Days

We will use pandas' internal plotting methods to create a scatter plot of rain over the days, omitting the

In [9]:

```
df.plot.scatter(x='date', y='rain', xlabel='date', ylabel='Rain (mm)', title='Rain over t
plt.xticks([])
plt.show()
```

Group Data by Seasons

We will group the data by seasons (Winter, Spring, Summer, Autumn) by associating the months of the year and then plot a boxplot of rain in the four seasons using seaborn.

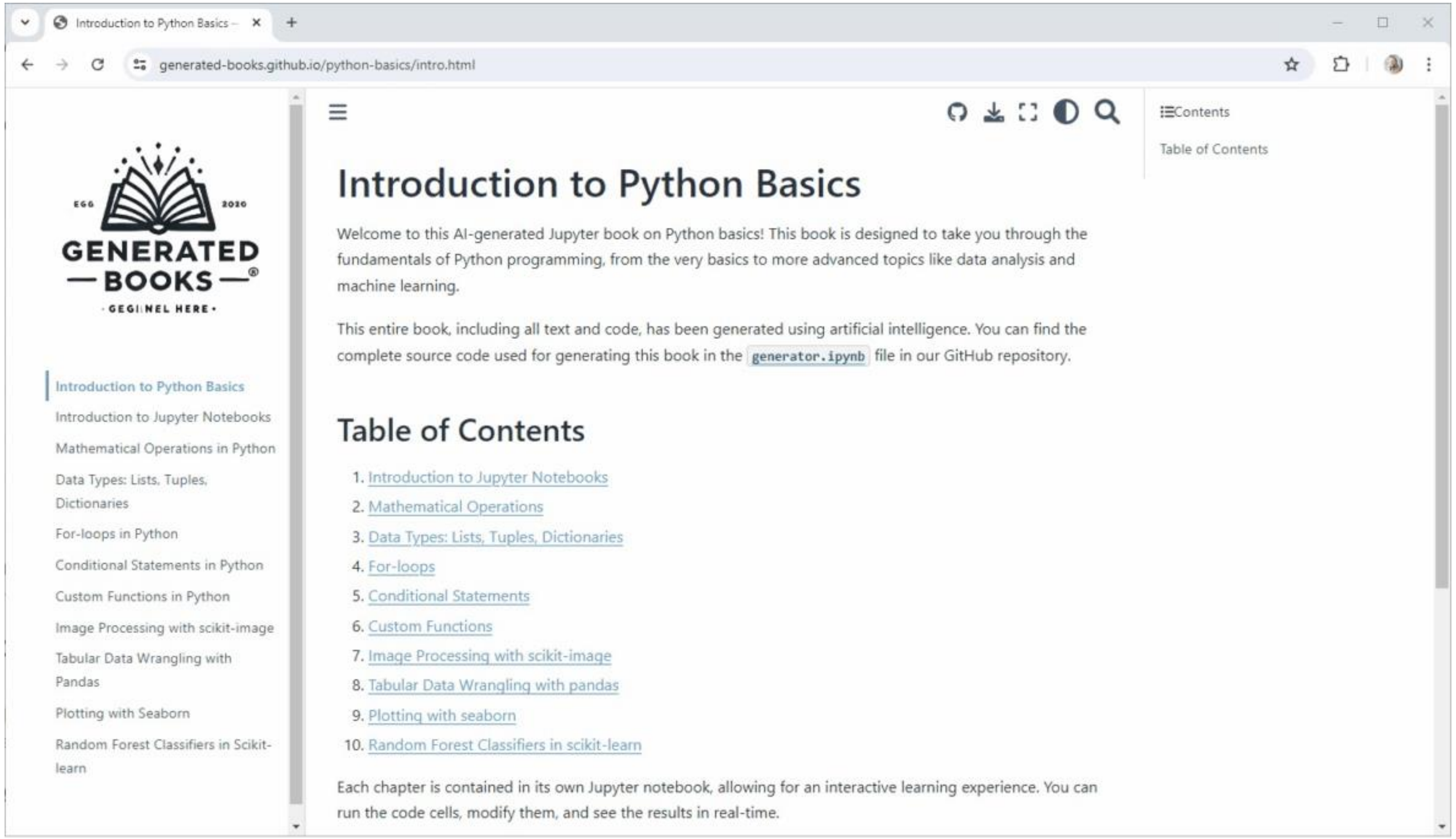
In [8]:

```
def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Autumn'

df['season'] = pd.to_datetime(df['date']).dt.month.apply(get_season)
sns.boxplot(x='season', y='rain', data=df)
plt.title('Rain by Season')
plt.xlabel('Season')
plt.ylabel('Rain (mm)')
plt.show()
```

Cropped, licensed CC-BY 4.0 by Matthias Täschner & Robert Haase
https://github.com/ScaDS/Python-training-2024/blob/main/2_tables_plots/python_weather_analysis.ipynb

Generating books

… also great for learning Python


Introduction to Python Basics –
generated-books.github.io/python-basics/intro.html
EGG
2020
GENERATED
BOOKS
GEGINEL HERE
Contents
Table of Contents

- Introduction to Python Basics
- Introduction to Jupyter Notebooks
- Mathematical Operations in Python
- Data Types: Lists, Tuples, Dictionaries
- For-loops in Python
- Conditional Statements in Python
- Custom Functions in Python
- Image Processing with scikit-image
- Tabular Data Wrangling with Pandas
- Plotting with Seaborn
- Random Forest Classifiers in Scikit-learn

Introduction to Python Basics

Welcome to this AI-generated Jupyter book on Python basics! This book is designed to take you through the fundamentals of Python programming, from the very basics to more advanced topics like data analysis and machine learning.

This entire book, including all text and code, has been generated using artificial intelligence. You can find the complete source code used for generating this book in the `generator.ipynb` file in our GitHub repository.

Table of Contents

1. Introduction to Jupyter Notebooks
2. Mathematical Operations
3. Data Types: Lists, Tuples, Dictionaries
4. For-loops
5. Conditional Statements
6. Custom Functions
7. Image Processing with scikit-image
8. Tabular Data Wrangling with pandas
9. Plotting with seaborn
10. Random Forest Classifiers in scikit-learn

Each chapter is contained in its own Jupyter notebook, allowing for an interactive learning experience. You can run the code cells, modify them, and see the results in real-time.

100% LLM-generated content

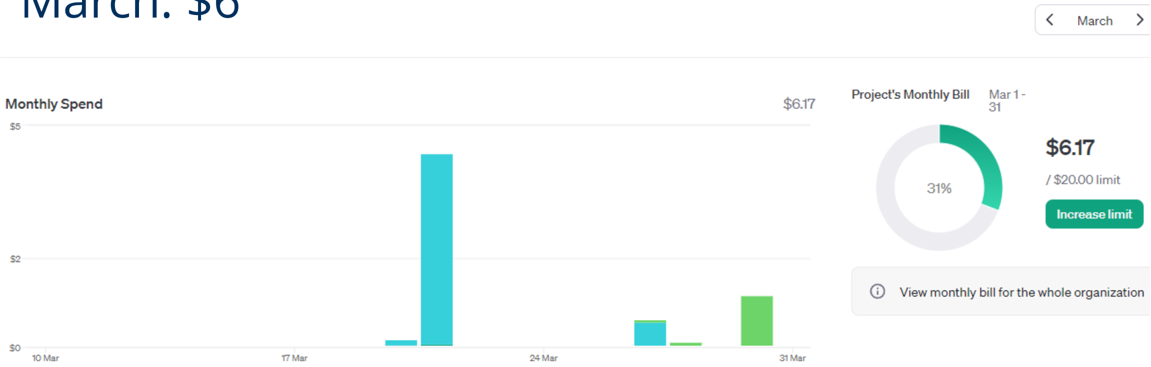
Costs:
- 1 day of prompt engineering
- $3.90 API usage of Claude 3.5 Sonnet (Anthropic)

https://github.com/generated-books/python-basics

Costs

I consider myself an advanced LLM user. I mostly use services from OpenAI / chatGPT. My recent monthly usage:

March: $6

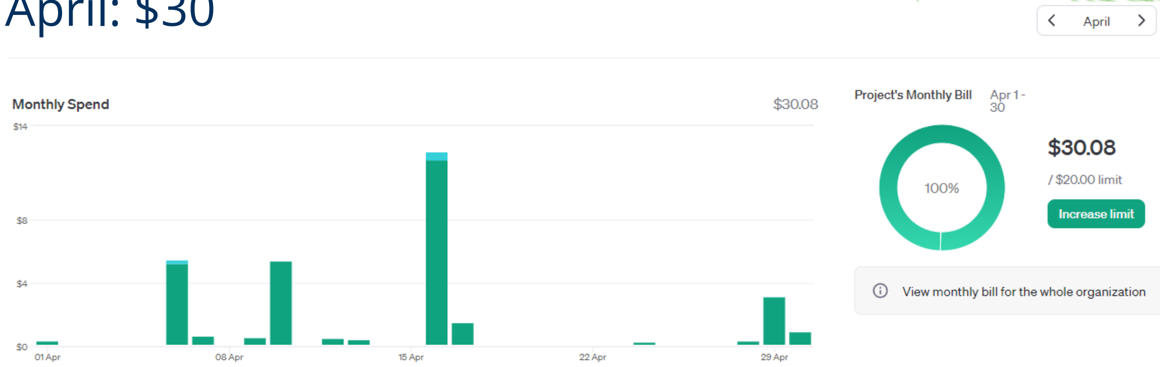


Project's Monthly Bill Mar 1 - 31

$6.17 / $20.00 limit

ⓘ View monthly bill for the whole organization

April: $30

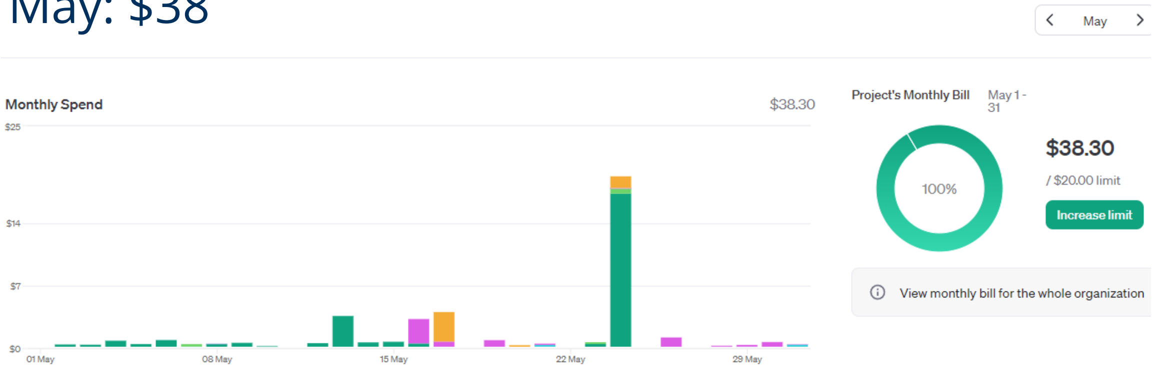


Project's Monthly Bill Apr 1 - 30

$30.08 / $20.00 limit

ⓘ View monthly bill for the whole organization

May: $38

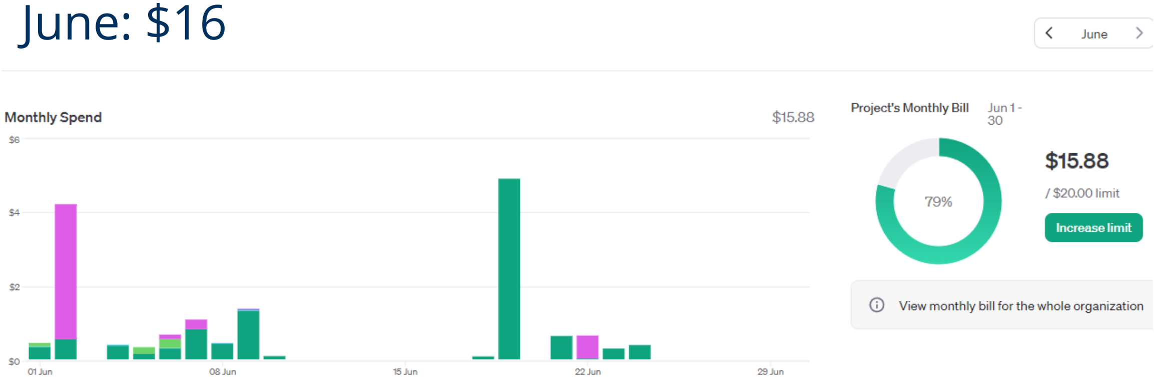


Project's Monthly Bill May 1 - 31

$38.30 / $20.00 limit

ⓘ View monthly bill for the whole organization

June: $16



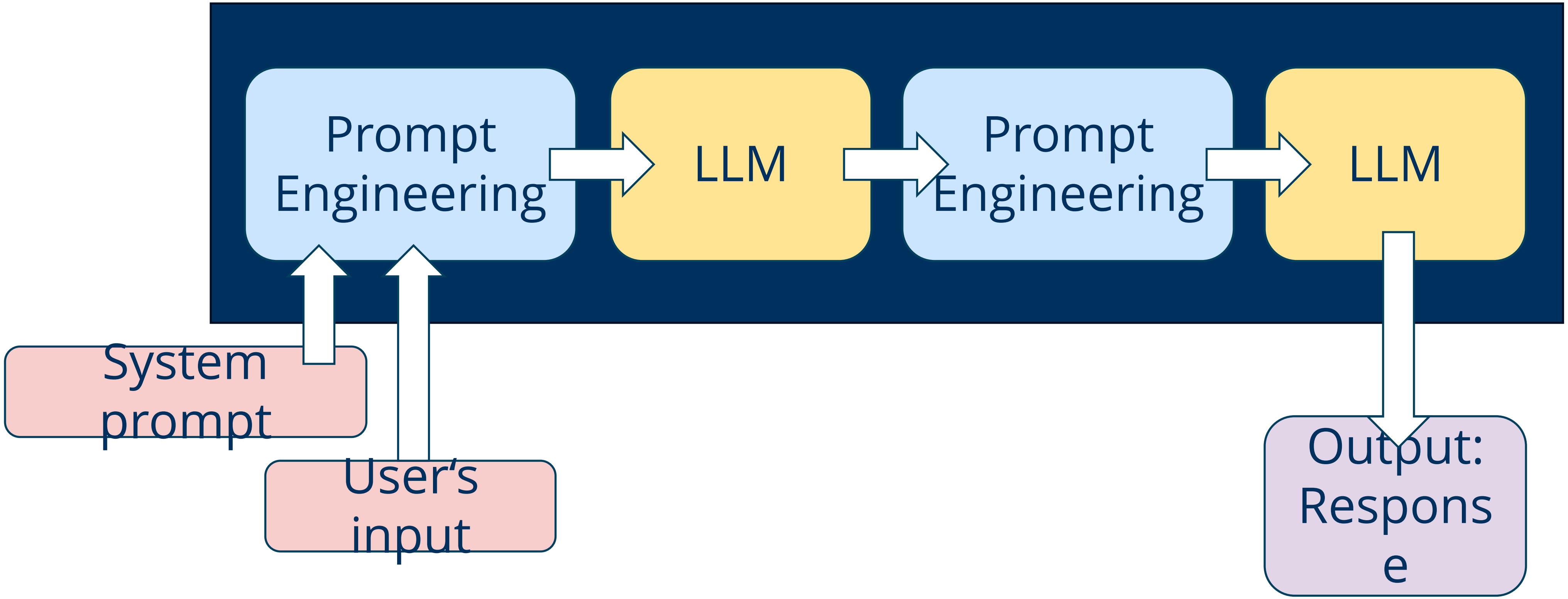
Project's Monthly Bill Jun 1 - 30

$15.88 / $20.00 limit

ⓘ View monthly bill for the whole organization

Prompt Engineering

Integrating / combining LLMs with other technology



Prompt Engineering

LLM

Prompt Engineering

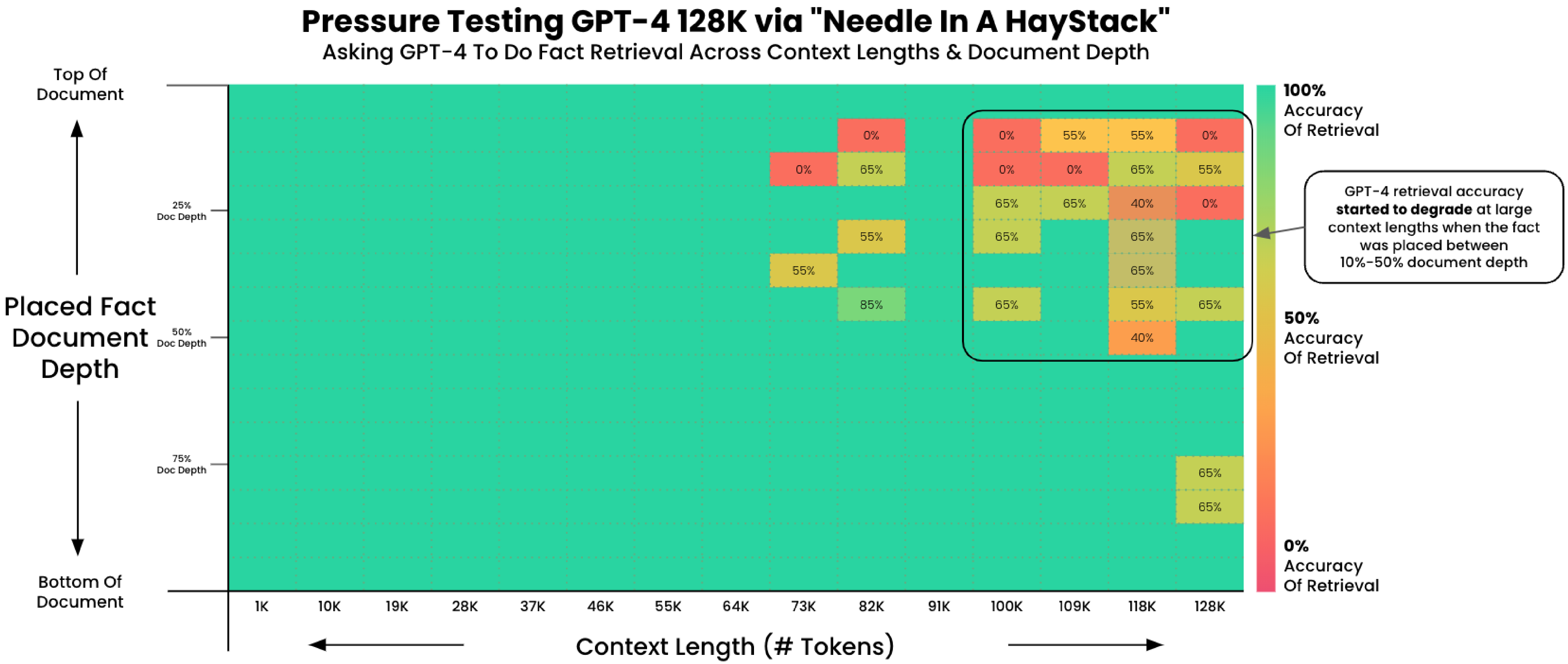
LLM

System prompt

User's input

Output: Respons e

Long contexts



Pressure Testing GPT-4 128K via "Needle In A HayStack"

Asking GPT-4 To Do Fact Retrieval Across Context Lengths & Document Depth

GPT-4 retrieval accuracy **started to degrade** at large context lengths when the fact was placed between 10%-50% document depth

Visualization cropped from Greg Kamradt, Licensed MIT (modified),
Source: https://github.com/gkamradt/LLMTest_NeedleInAHaystack

Under the hood: prompt engineering

Context-dependent system prompt considering

- local variables and functions
- installed python libraries
- chat history

+ your prompt

```
You are a extremely talented bioimage analyst and you use
Python to solve your tasks …
```

…

~~~
## Python specific code snippets
If the user asks for those simple tasks, use these code snippets.

   * Load an image file from disc and store it in a variable:
   ```
   from skimage.io import imread
   image = imread(filename)
   ```
~~~

…

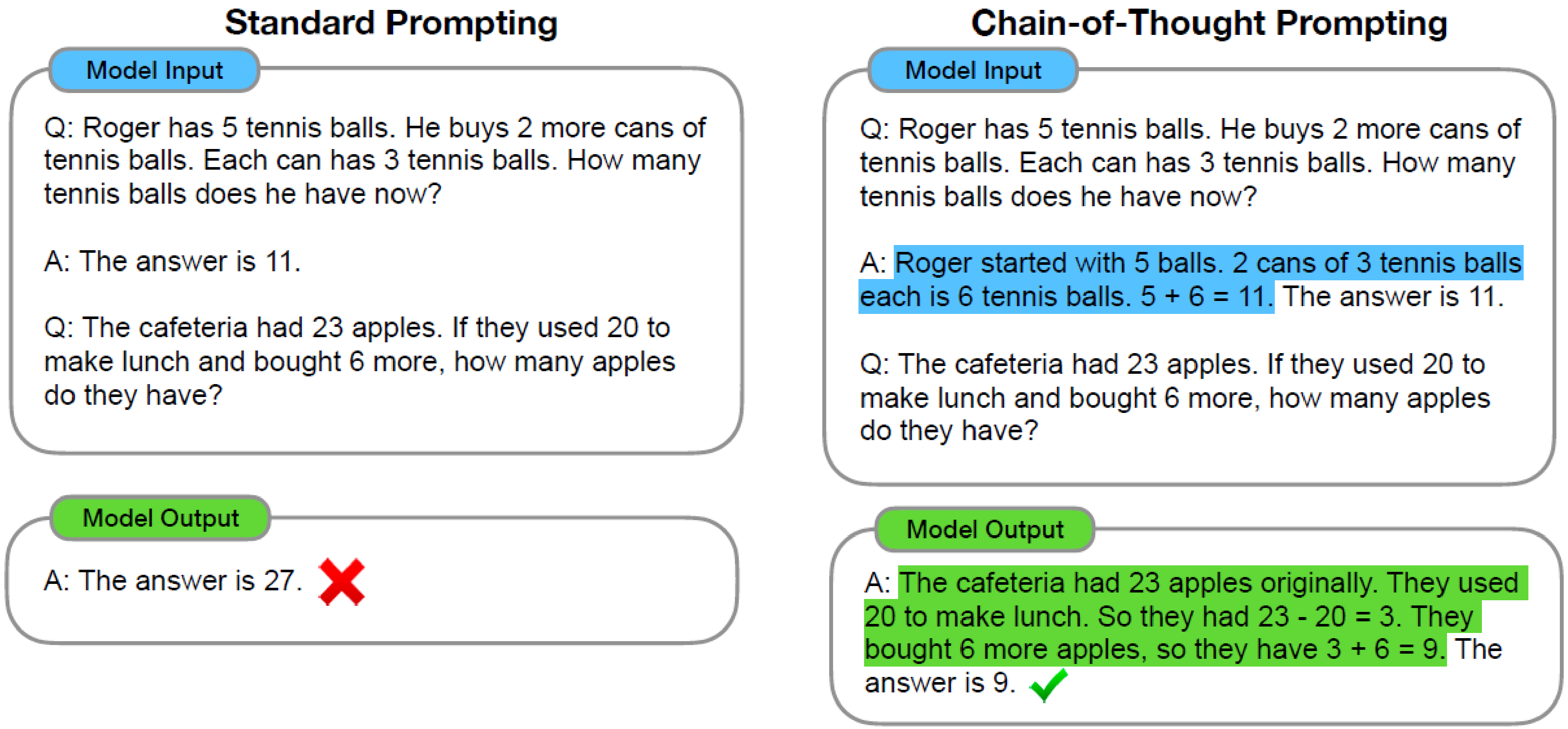
```
## Todos
Answer your response in three sections:
   1. Summary: First provide a short summary of the task.
   2. Plan: Provide a concise step-by-step plan without any code.
   3. Code: Provide the code.
```

About 6500 tokens (≈words)

Chain-of-throught prompting

Demonstrating reasoning steps to the model

„Let‘s think this step-by-step.“



Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

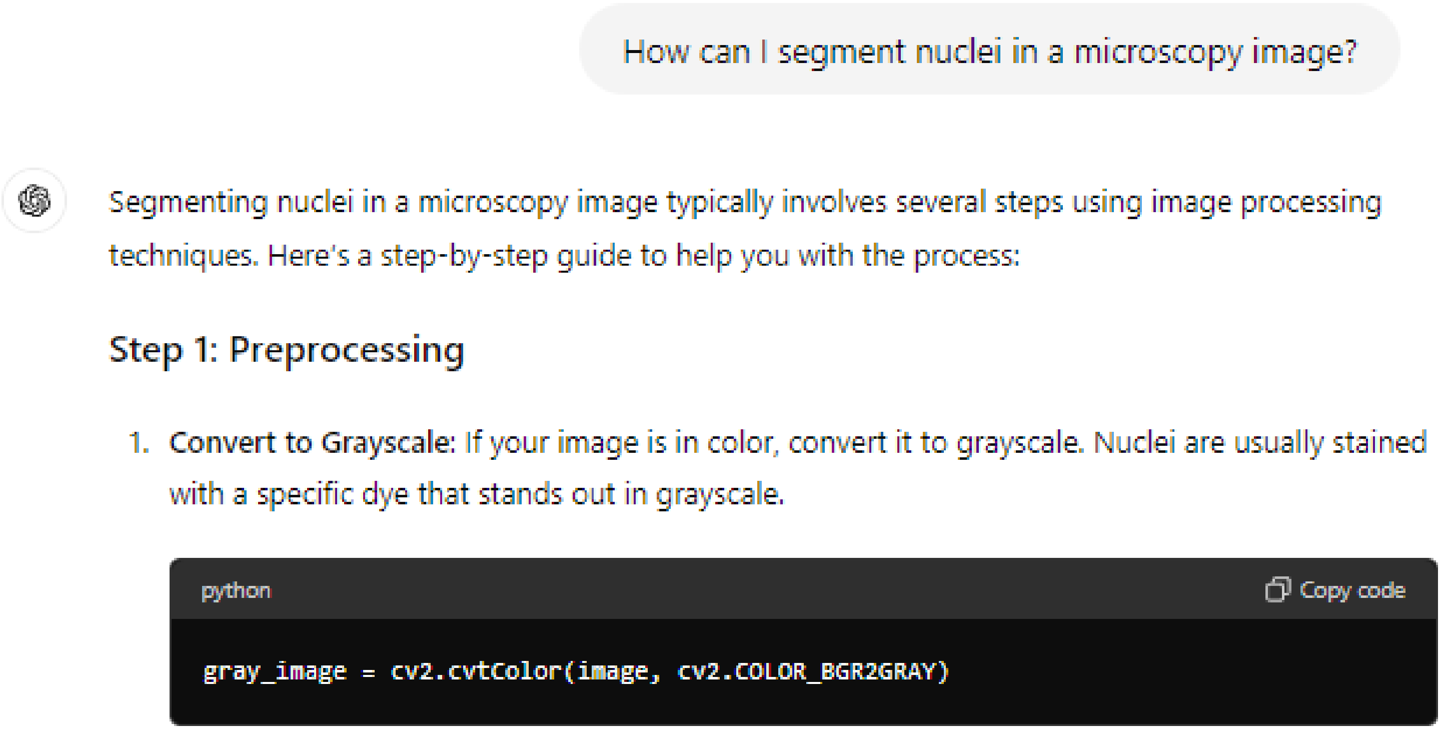
A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Cropped from Wei et al 2022, licensed CC-BY 4.0 https://arxiv.org/abs/2201.11903

Chain-of-throught prompting

Example: ChatGPT

„Let‘s think this step-by-step“ seems part of the system-prompt



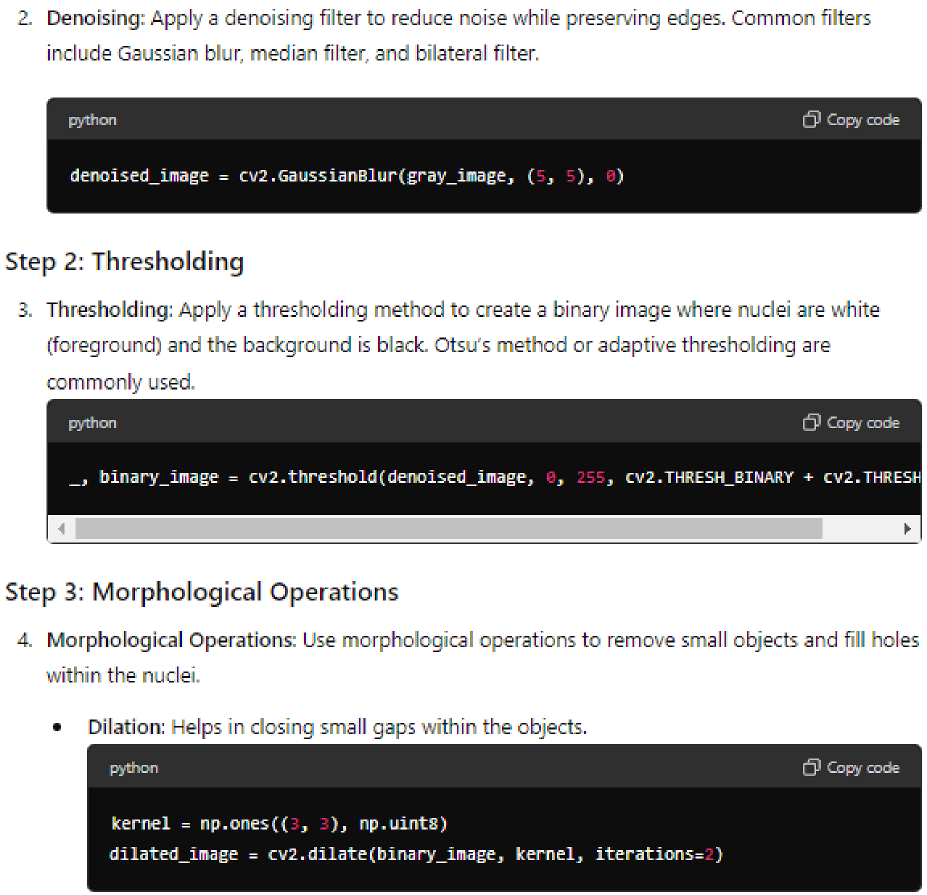
How can I segment nuclei in a microscopy image?

Segmenting nuclei in a microscopy image typically involves several steps using image processing techniques. Here's a step-by-step guide to help you with the process:

Step 1: Preprocessing

1. **Convert to Grayscale:** If your image is in color, convert it to grayscale. Nuclei are usually stained with a specific dye that stands out in grayscale.

```python
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```



2. **Denoising:** Apply a denoising filter to reduce noise while preserving edges. Common filters include Gaussian blur, median filter, and bilateral filter.

```python
denoised_image = cv2.GaussianBlur(gray_image, (5, 5), 0)
```

Step 2: Thresholding

3. **Thresholding:** Apply a thresholding method to create a binary image where nuclei are white (foreground) and the background is black. Otsu's method or adaptive thresholding are commonly used.

```python
_, binary_image = cv2.threshold(denoised_image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH
```

Step 3: Morphological Operations

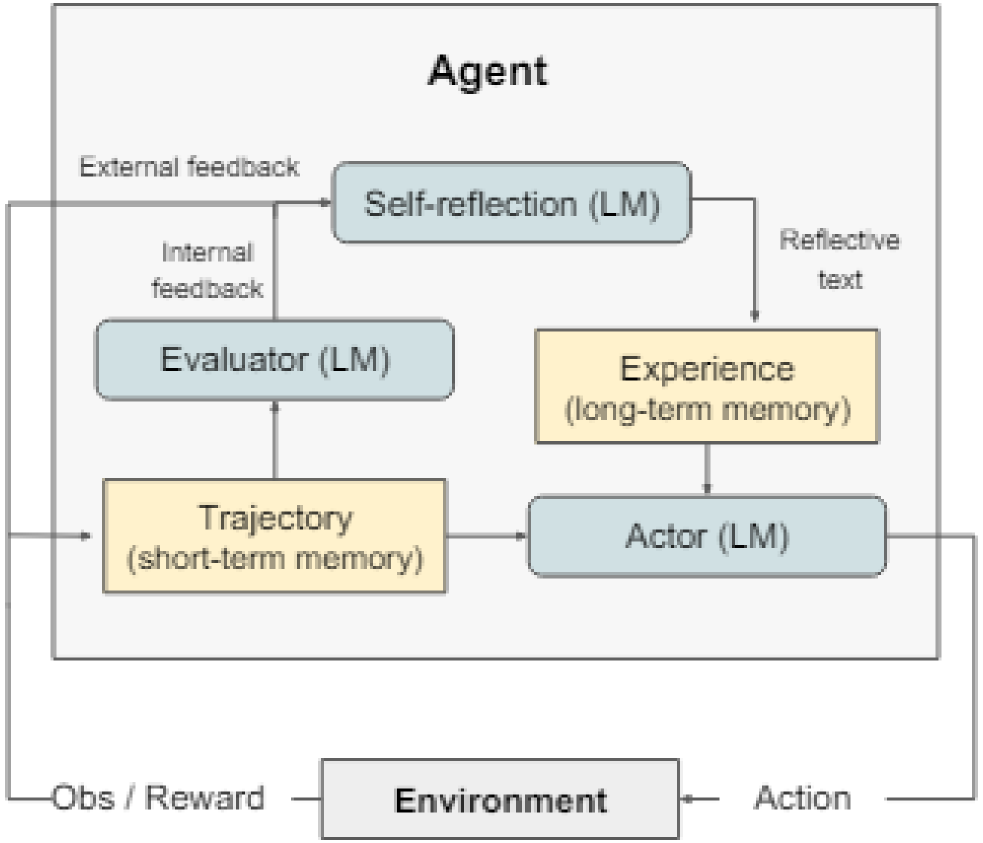
4. **Morphological Operations:** Use morphological operations to remove small objects and fill holes within the nuclei.

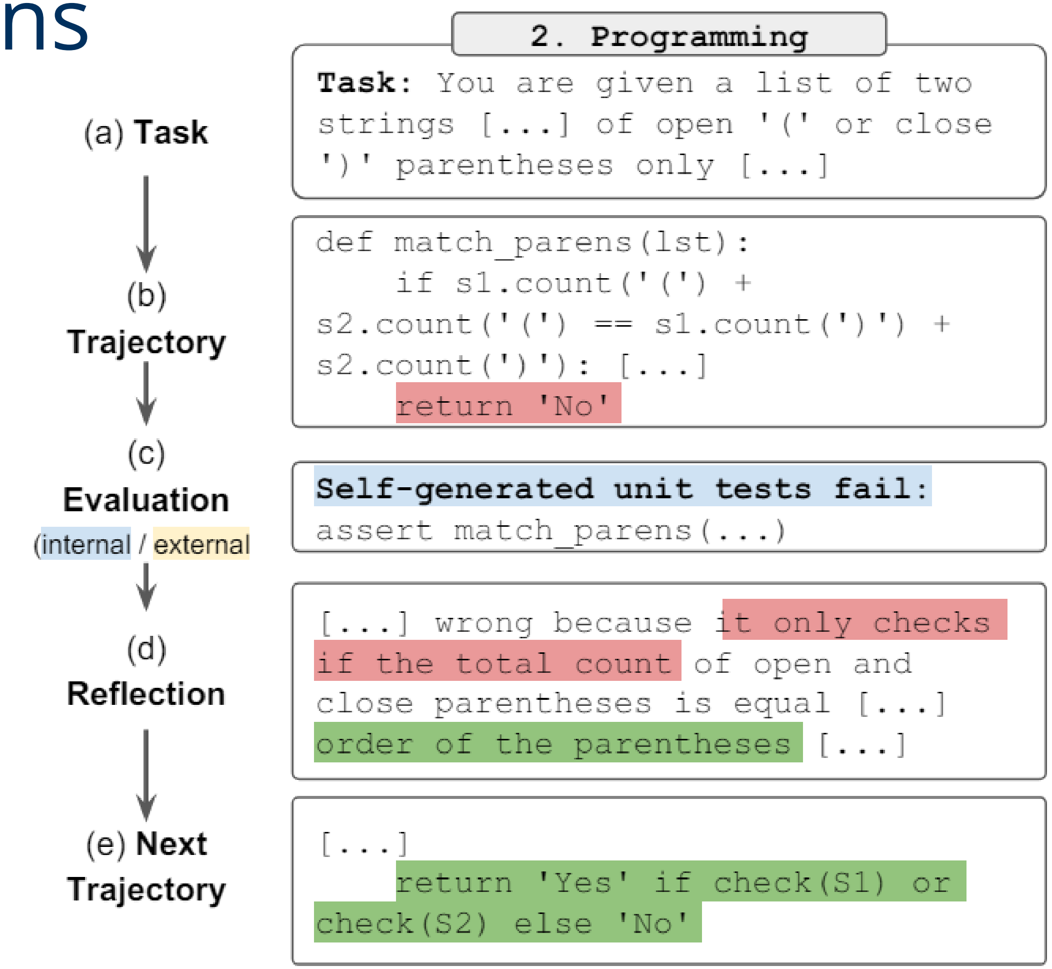
- **Dilation:** Helps in closing small gaps within the objects.

```python
kernel = np.ones((3, 3), np.uint8)
dilated_image = cv2.dilate(binary_image, kernel, iterations=2)
```

Reflection

Iterating over tasks/solutions





Task: You are given a list of two strings [...] of open '(' or close ')' parentheses only [...]

```
def match_parens(lst):
    if s1.count('(') +
s2.count('(') == s1.count(')') +
s2.count(')'): [...]
    return 'No'
```

Self-generated unit tests fail:
`assert match_parens(...)`

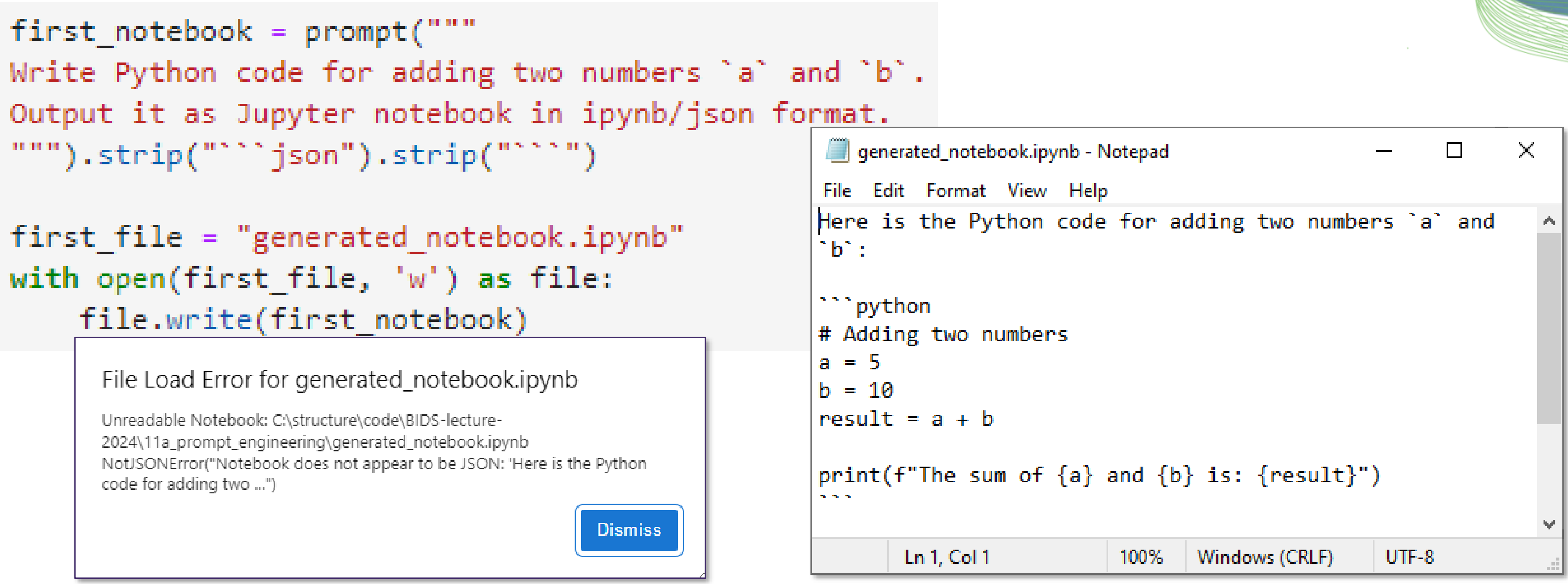
[...] wrong because it only checks if the total count of open and close parentheses is equal [...] order of the parentheses [...]

```
[...]
    return 'Yes' if check(S1) or
check(S2) else 'No'
```

Cropped from Shinn et al 2023, licensed CC-BY 4.0 https://arxiv.org/abs/2303.11366

Reflection

Example task: Generate a Jupyter notebook



```
first_notebook = prompt("""
Write Python code for adding two numbers `a` and `b`.
Output it as Jupyter notebook in ipynb/json format.
""").strip("```json").strip("```")

first_file = "generated_notebook.ipynb"
with open(first_file, 'w') as file:
    file.write(first_notebook)
```

File Load Error for generated_notebook.ipynb

Unreadable Notebook: C:\structure\code\BIDS-lecture-2024\11a_prompt_engineering\generated_notebook.ipynb NotJSONError("Notebook does not appear to be JSON: 'Here is the Python code for adding two ...")

generated_notebook.ipynb - Notepad
File Edit Format View Help

~~~~
Here is the Python code for adding two numbers `a` and
`b`:

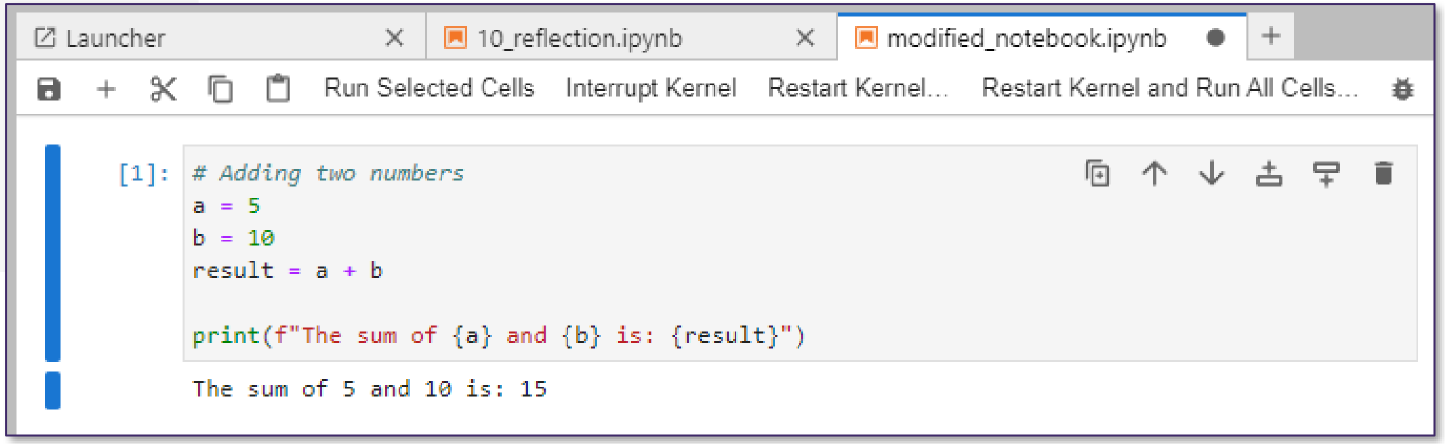
```python
# Adding two numbers
a = 5
b = 10
result = a + b

print(f"The sum of {a} and {b} is: {result}")
```
~~~~

https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11a_prompt_engineering/10_reflection.ipynb

Reflection

Example task: Generate a Jupyter notebook

```python
second_notebook = prompt(f"""
Take the following text and extract the Jupyter
notebook ipynb/json from it:

{first_notebook}

Make sure the output is in ipynb/json format.
""").strip("```json").strip("```")

second_file = "modified_notebook.ipynb"
with open(second_file, 'w') as file:
    file.write(second_notebook)
```



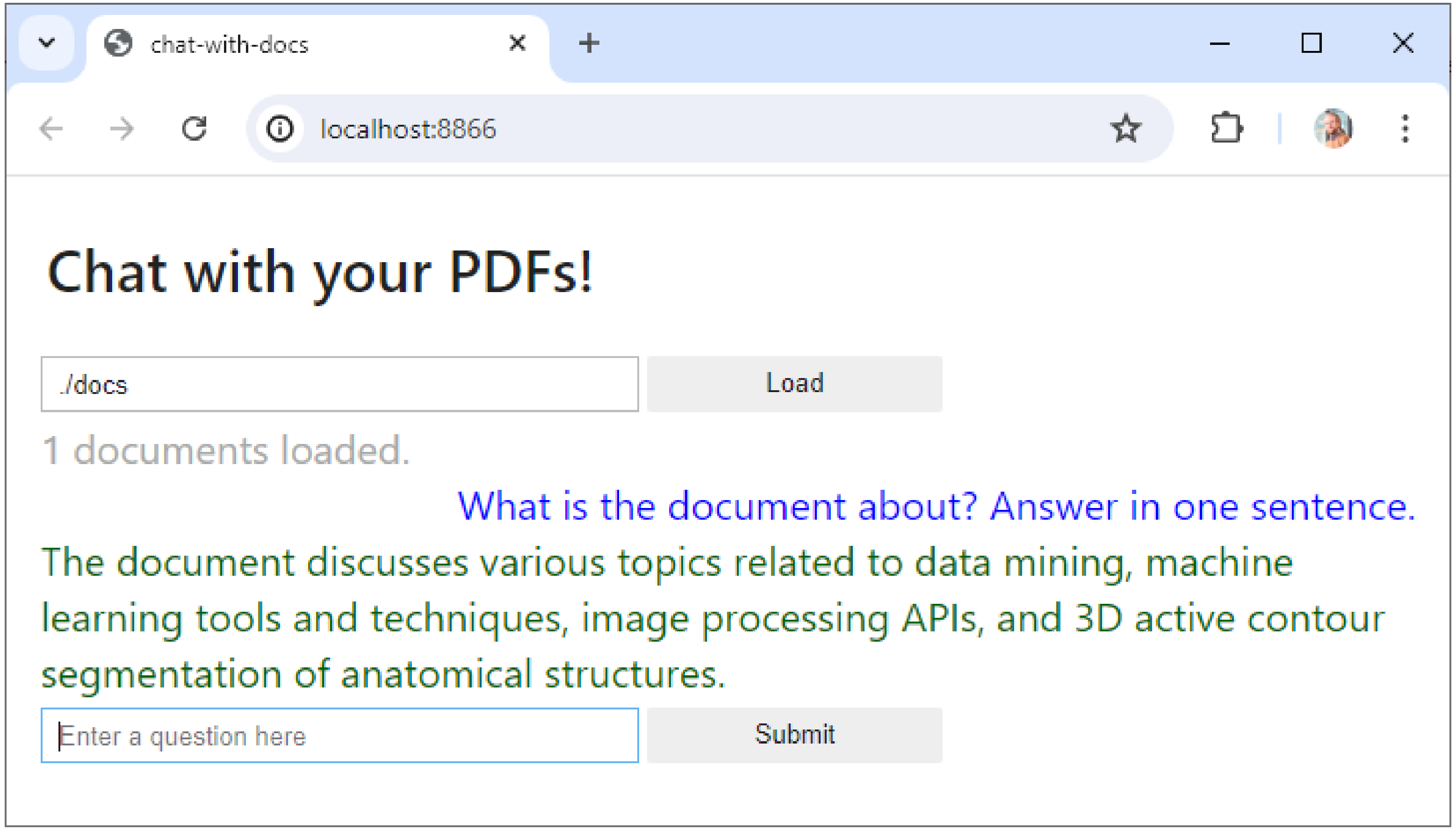
```python
# Adding two numbers
a = 5
b = 10
result = a + b

print(f"The sum of {a} and {b} is: {result}")
```

```
The sum of 5 and 10 is: 15
```

https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11a_prompt_engineering/10_reflection.ipynb

Retrieval augmented generation

… the technology behind various chatbots:



Chat with your PDFs!

1 documents loaded.

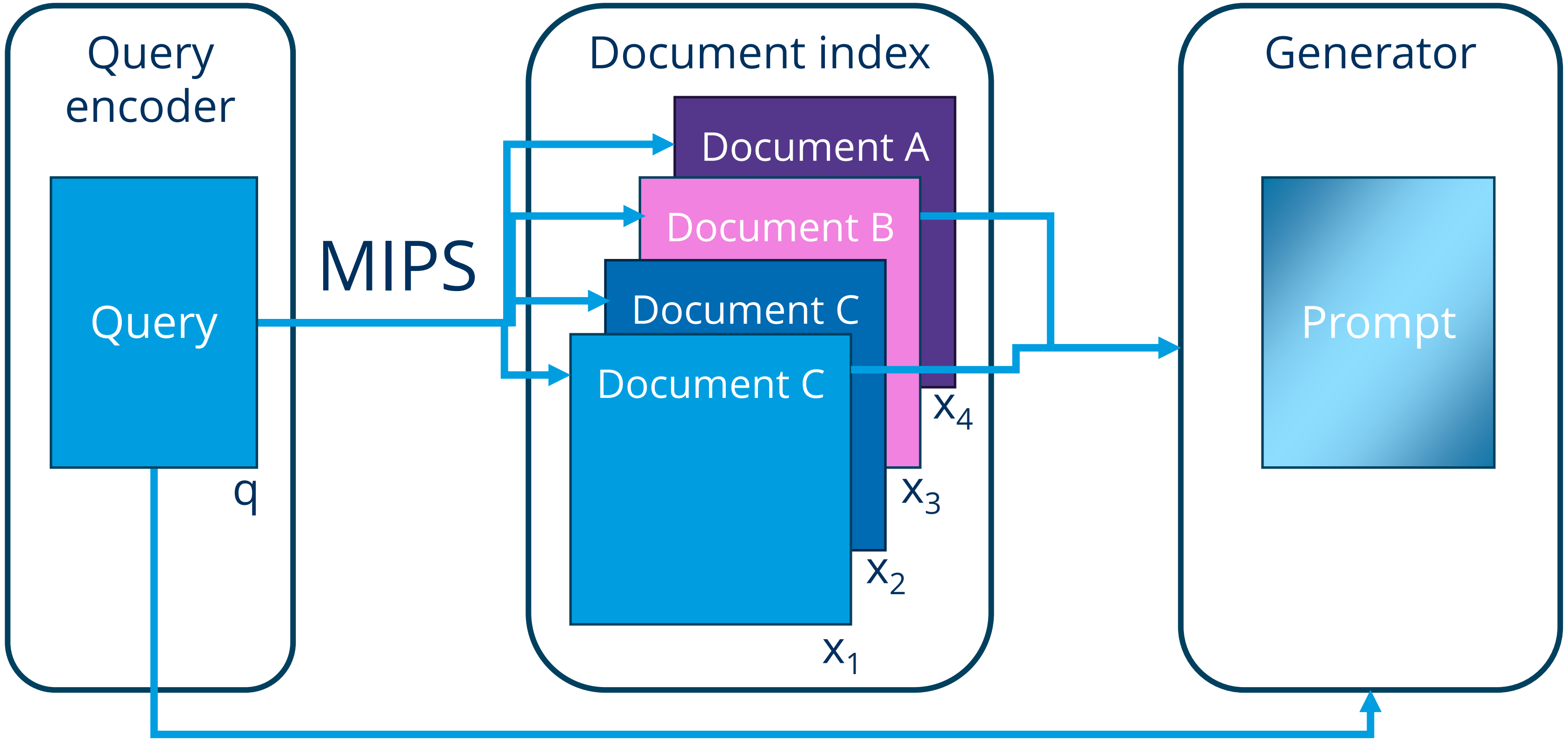
What is the document about? Answer in one sentence.

The document discusses various topics related to data mining, machine learning tools and techniques, image processing APIs, and 3D active contour segmentation of anatomical structures.

https://github.com/haesleinhuepf/simple-chat-with-docs/

Retrieval Augmented Generation

Enriching a prompt with relevant context



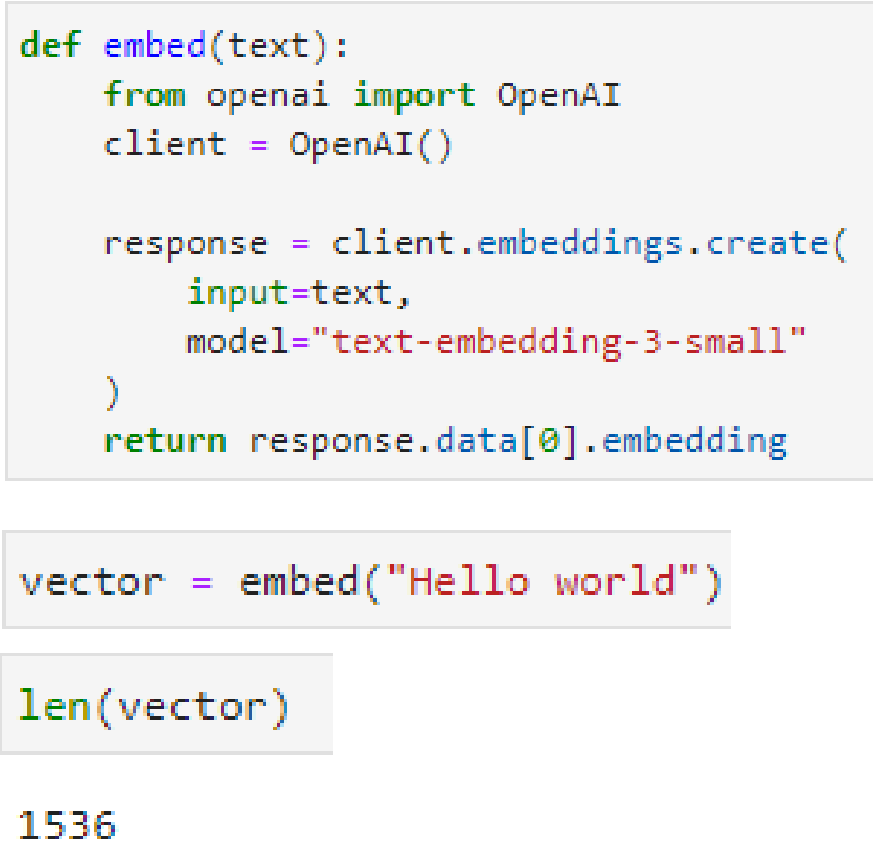
Maximum inner product search

$$x = \text{argmax}_{x_i \in D} x_i^T q$$

Read more: Lewis et al 2020
https://arxiv.org/abs/2005.11401

Retrieval Augmented Generation

Embeddings

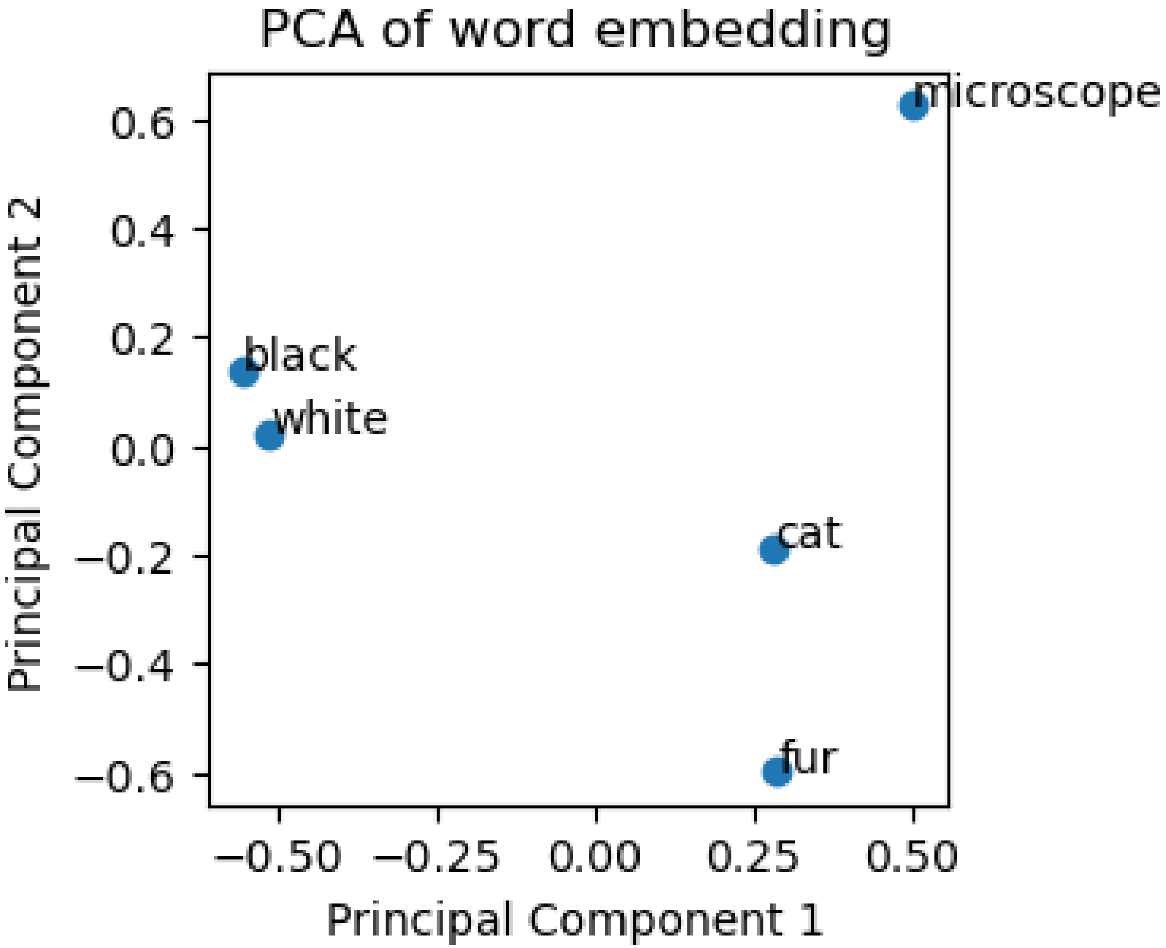


```
def embed(text):
    from openai import OpenAI
    client = OpenAI()

    response = client.embeddings.create(
        input=text,
        model="text-embedding-3-small"
    )
    return response.data[0].embedding
```

```
vector = embed("Hello world")
```

```
len(vector)
```

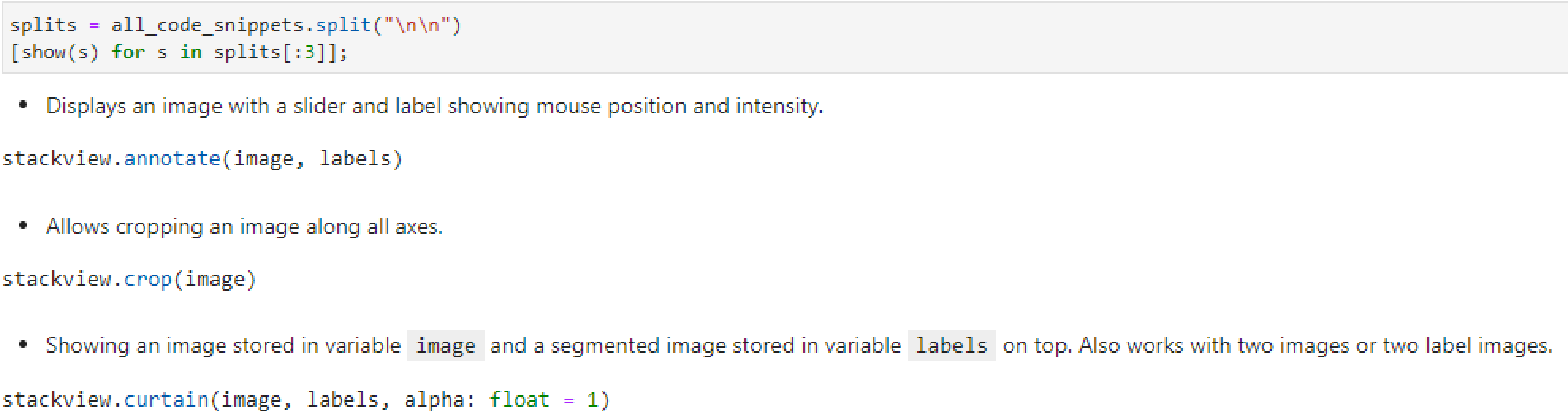
```
1536
```



https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11c_rag/10_word-embeddings.ipynb

Retrieval augmented generation

0. Encode the knowledge base (code snippets)



```
splits = all_code_snippets.split("\n\n")
[show(s) for s in splits[:3]];
```

- Displays an image with a slider and label showing mouse position and intensity.

```
stackview.annotate(image, labels)
```

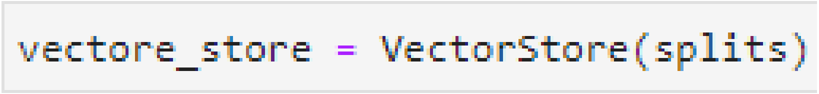
- Allows cropping an image along all axes.

```
stackview.crop(image)
```

- Showing an image stored in variable `image` and a segmented image stored in variable `labels` on top. Also works with two images or two label images.

```
stackview.curtain(image, labels, alpha: float = 1)
```

…



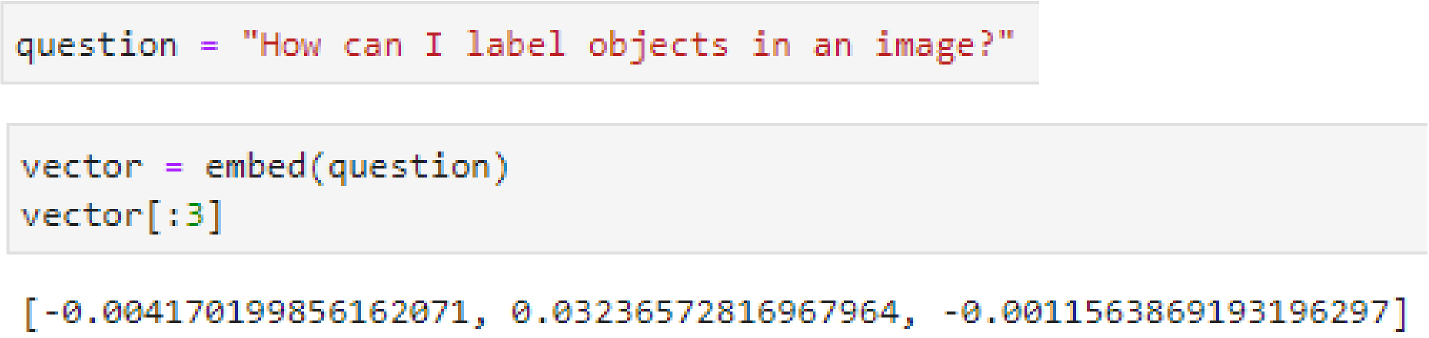
```
vectore_store = VectorStore(splits)
```

Embedding, ideally permanently stored!

https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11c_rag/20-simple-rag.ipynb

Retrieval augmented generation

1. Encode the question



```
question = "How can I label objects in an image?"
```

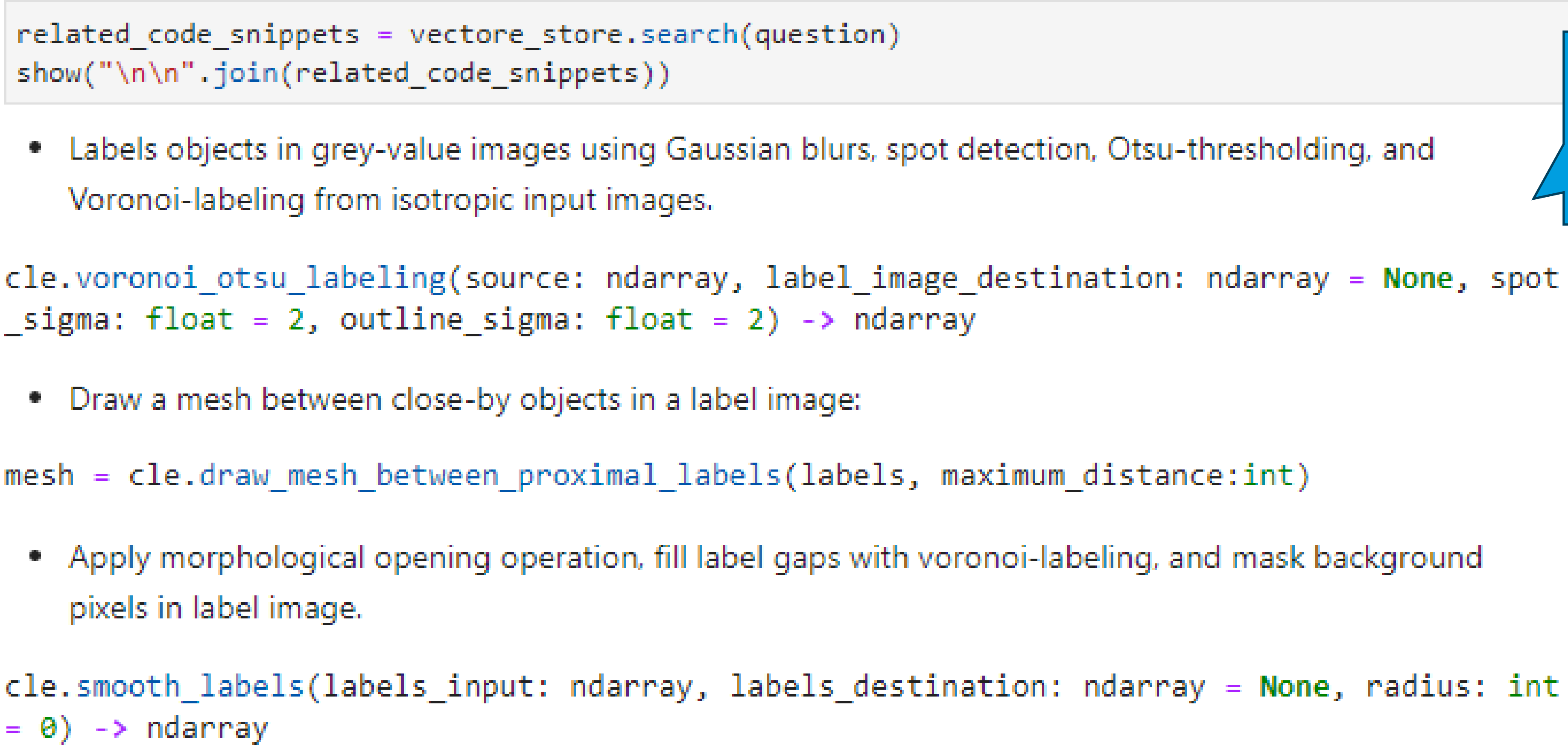
```
vector = embed(question)
vector[:3]
```

```
[-0.004170199856162071, 0.032365728169679 64, -0.0011563869193196297]
```

https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11c_rag/20-simple-rag.ipynb

Retrieval augmented generation

2. Identify related code-snippets



```
related_code_snippets = vectore_store.search(question)
show("\n\n".join(related_code_snippets))
```

- Labels objects in grey-value images using Gaussian blurs, spot detection, Otsu-thresholding, and Voronoi-labeling from isotropic input images.

```
cle.voronoi_otsu_labeling(source: ndarray, label_image_destination: ndarray = None, spot
_sigma: float = 2, outline_sigma: float = 2) -> ndarray
```

- Draw a mesh between close-by objects in a label image:

```
mesh = cle.draw_mesh_between_proximal_labels(labels, maximum_distance:int)
```

- Apply morphological opening operation, fill label gaps with voronoi-labeling, and mask background pixels in label image.

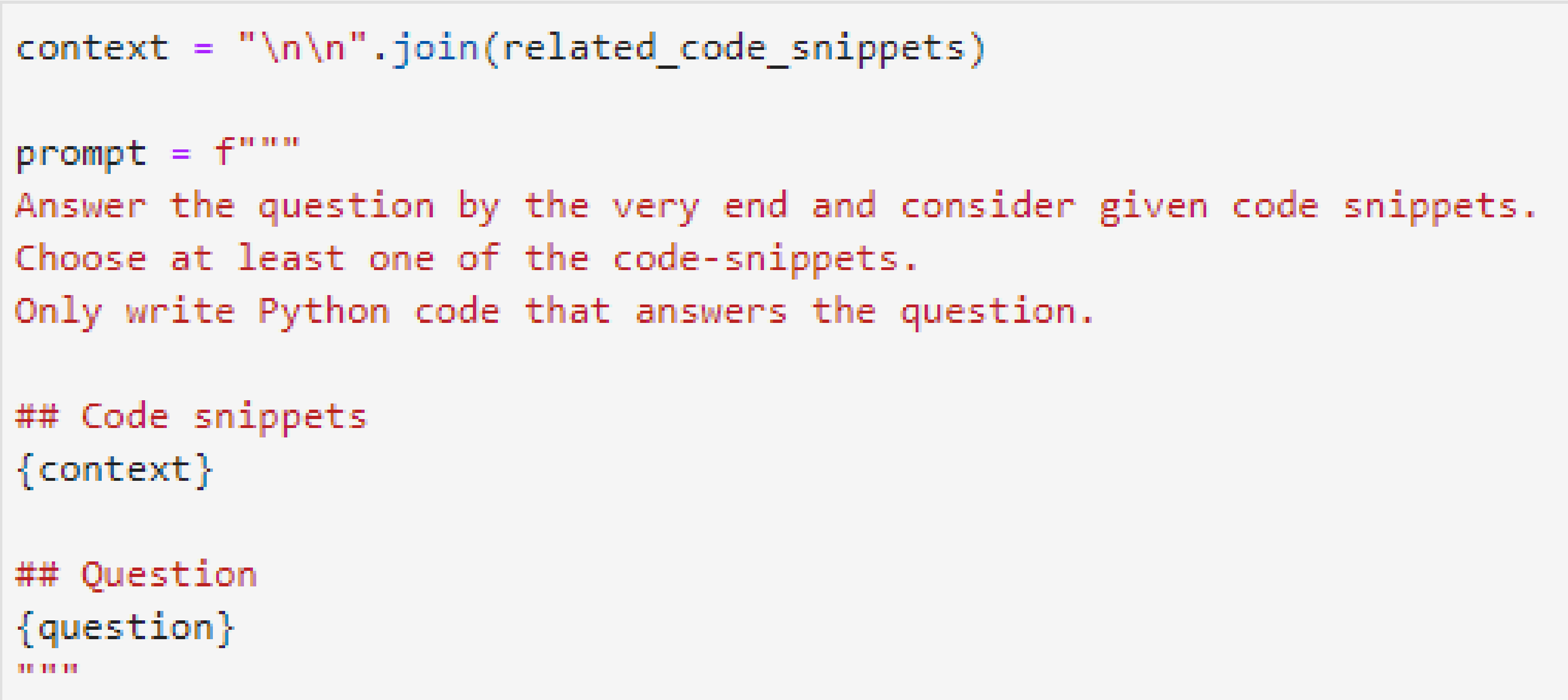
```
cle.smooth_labels(labels_input: ndarray, labels_destination: ndarray = None, radius: int
= 0) -> ndarray
```

Sorted by distance

https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11c_rag/20-simple-rag.ipynb

Retrieval augmented generation

3. Generate prompt

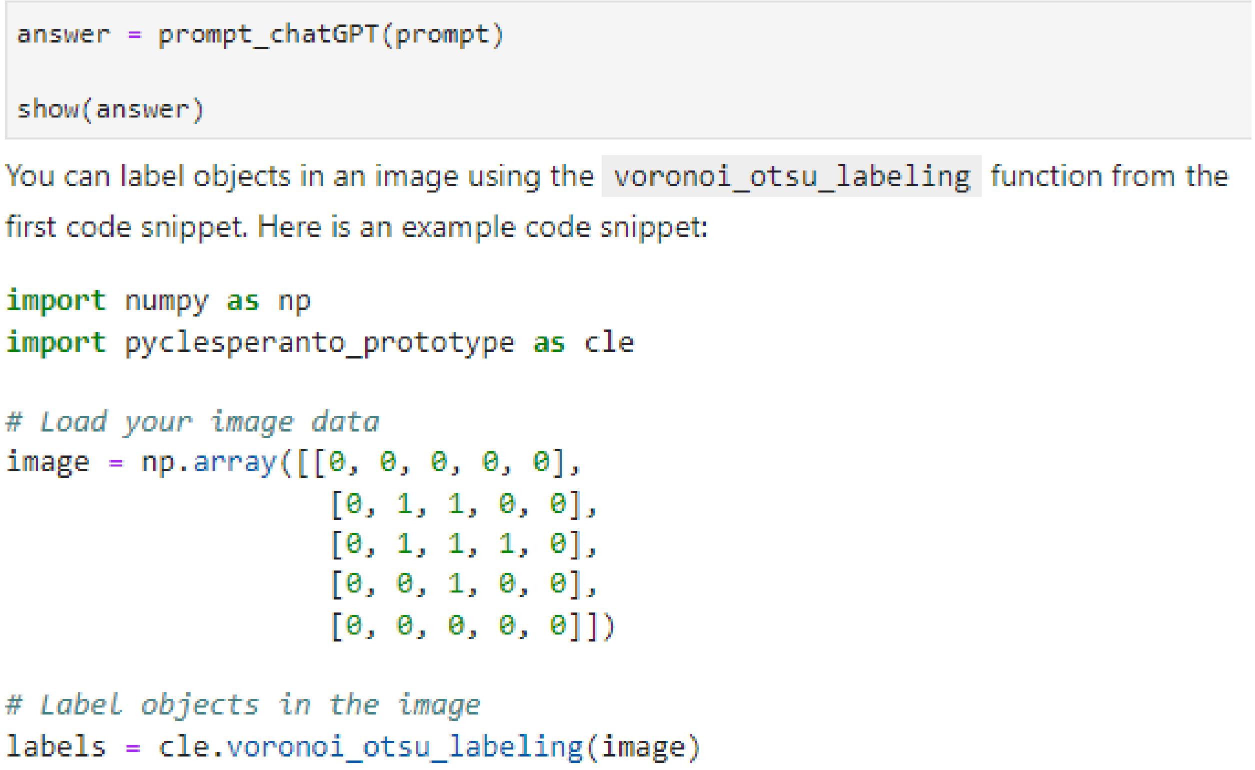


```
context = "\n\n".join(related_code_snippets)

prompt = f"""
Answer the question by the very end and consider given code snippets.
Choose at least one of the code-snippets.
Only write Python code that answers the question.

## Code snippets
{context}

## Question
{question}
"""
```

https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11c_rag/20-simple-rag.ipynb

Retrieval augmented generation

4. Retrieve answer



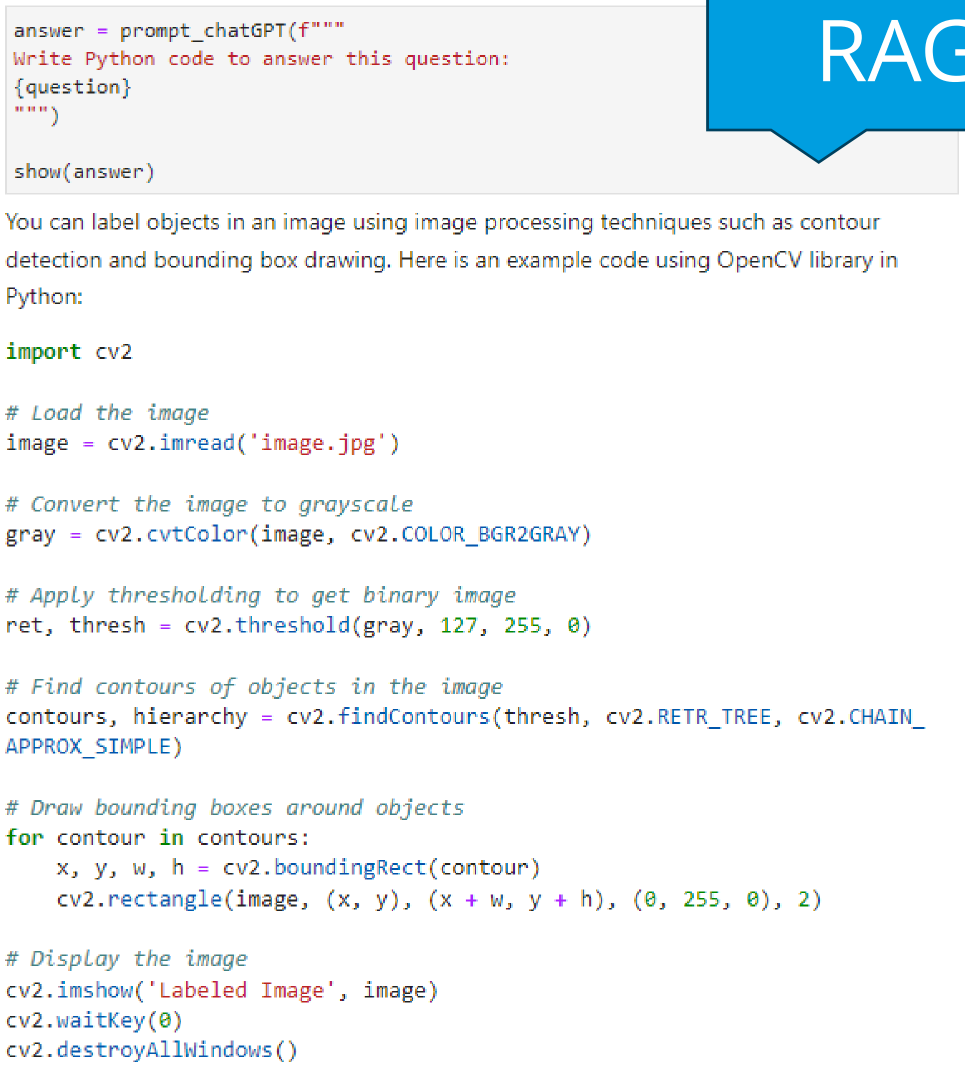
```
answer = prompt_chatGPT(prompt)

show(answer)
```

You can label objects in an image using the `voronoi_otsu_labeling` function from the first code snippet. Here is an example code snippet:

```
import numpy as np
import pyclesperanto_prototype as cle

# Load your image data
image = np.array([[0, 0, 0, 0, 0],
                  [0, 1, 1, 0, 0],
                  [0, 1, 1, 1, 0],
                  [0, 0, 1, 0, 0],
                  [0, 0, 0, 0, 0]])

# Label objects in the image
labels = cle.voronoi_otsu_labeling(image)
```

From our knowledge base

Without RAG



```
answer = prompt_chatGPT(f"""
Write Python code to answer this question:
{question}
""")

show(answer)
```

You can label objects in an image using image processing techniques such as contour detection and bounding box drawing. Here is an example code using OpenCV library in Python:

```
import cv2

# Load the image
image = cv2.imread('image.jpg')

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply thresholding to get binary image
ret, thresh = cv2.threshold(gray, 127, 255, 0)

# Find contours of objects in the image
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_
APPROX_SIMPLE)

# Draw bounding boxes around objects
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Display the image
cv2.imshow('Labeled Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

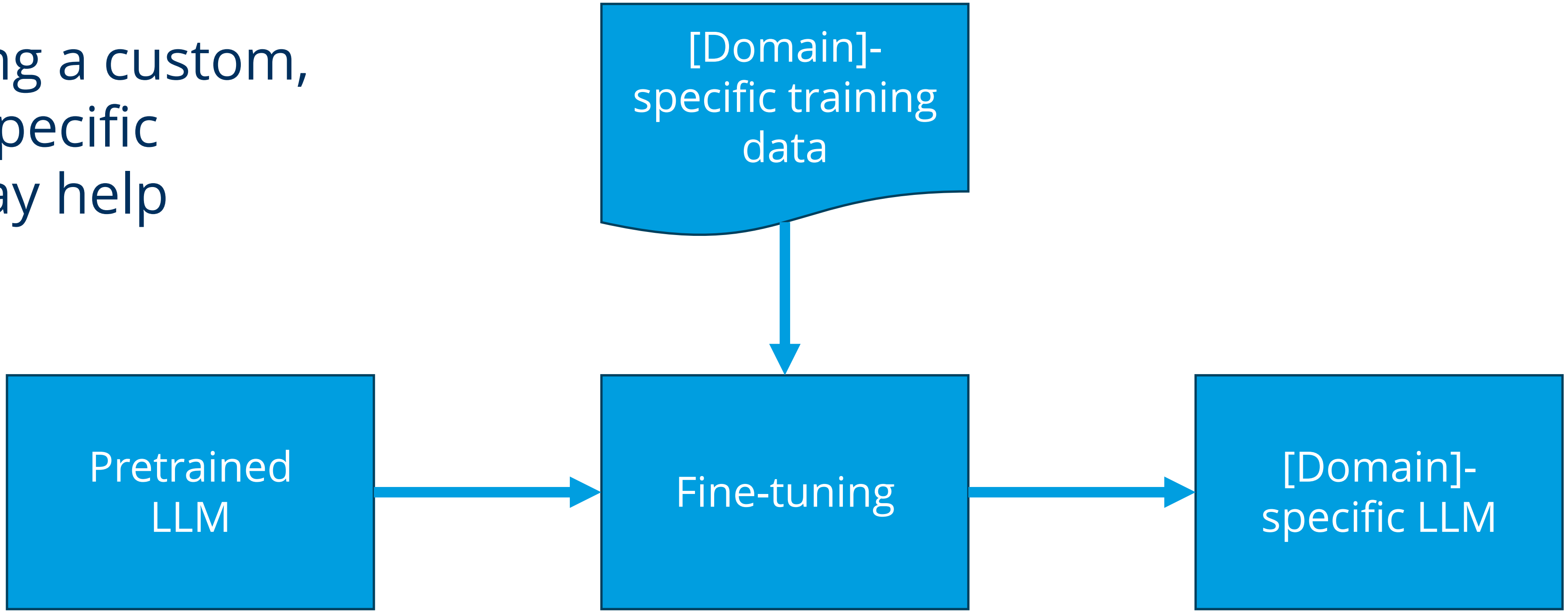
https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11c_rag/20-simple-rag.ipynb

Fine-tuning

Classical prompt-engineering is limited

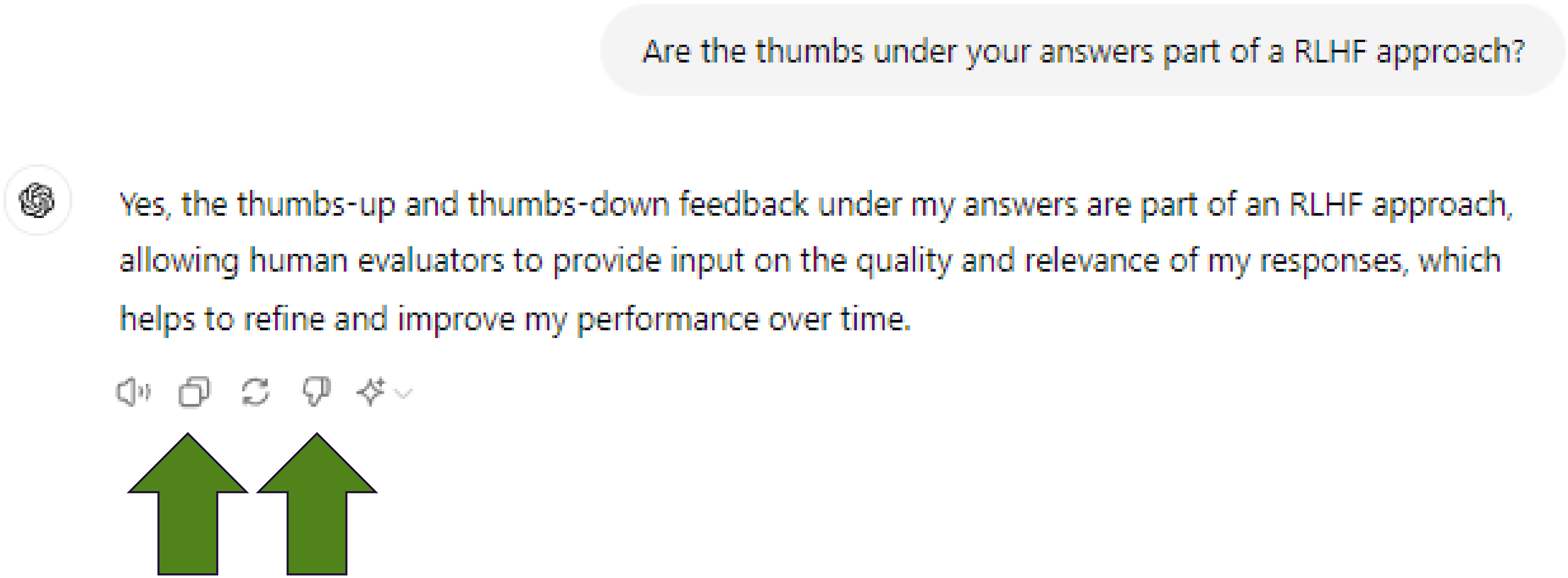
- Response time ↑
- Costs ↑

Fine-tuning a custom, domain-specific model may help



Fine-tuning

Reinforcement learning from human feedback (RLHF)



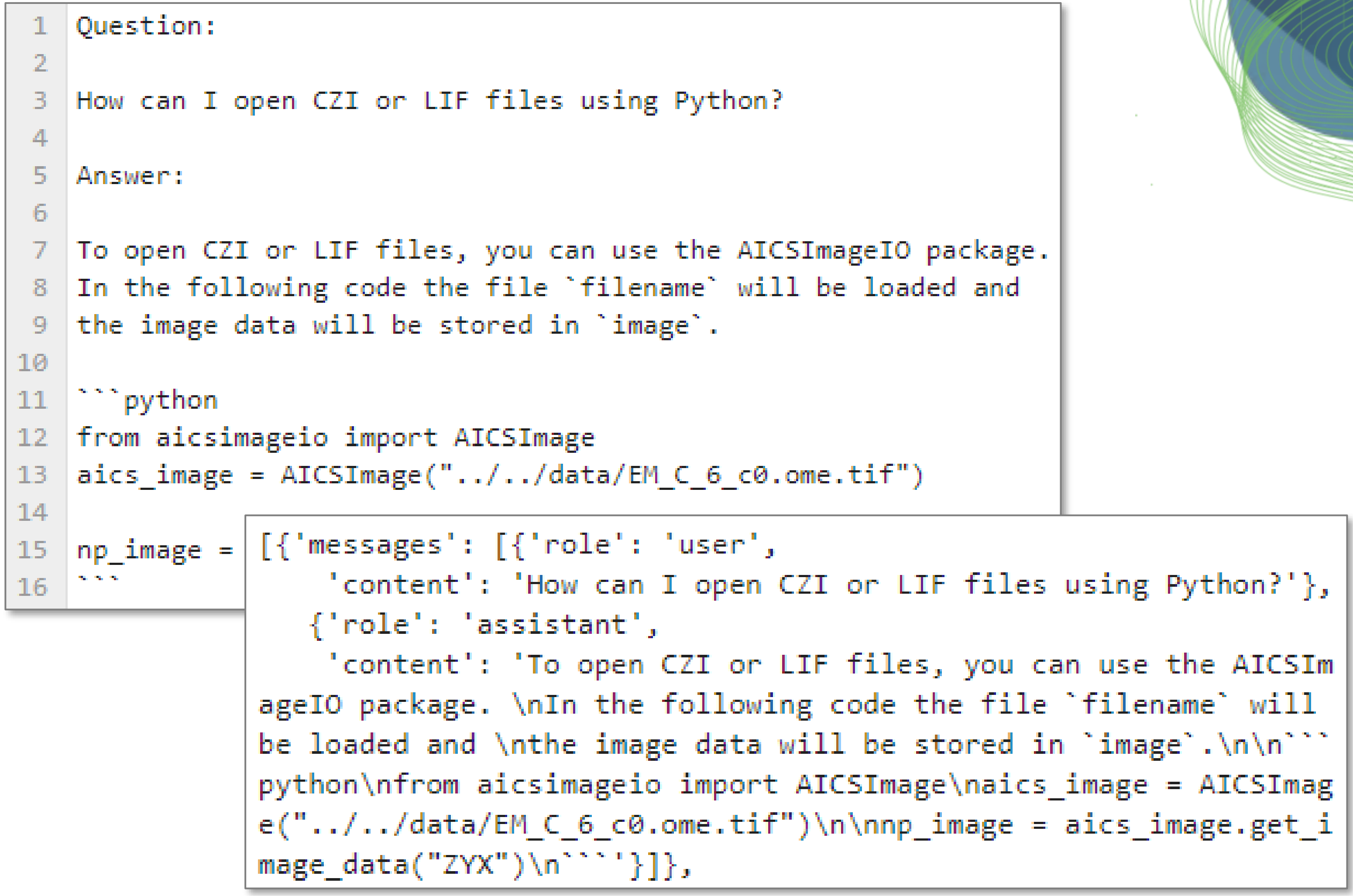
Are the thumbs under your answers part of a RLHF approach?

Yes, the thumbs-up and thumbs-down feedback under my answers are part of an RLHF approach, allowing human evaluators to provide input on the quality and relevance of my responses, which helps to refine and improve my performance over time.

Fine-tuning OpenAI's gpt-3.5-turbo

- Training data: conversations

- In JSON format



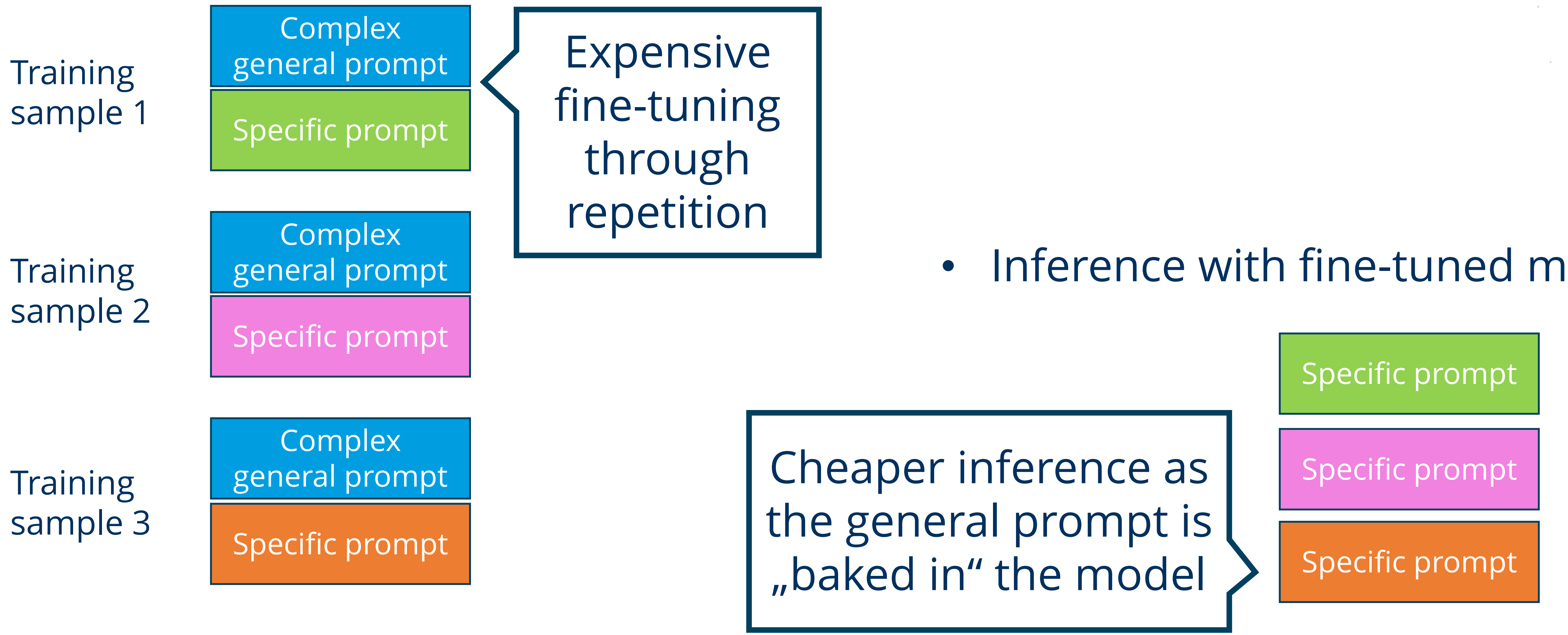
~~~
Question:

How can I open CZI or LIF files using Python?

Answer:

To open CZI or LIF files, you can use the AICSImageIO package.
In the following code the file `filename` will be loaded and
the image data will be stored in `image`.

```python
from aicsimageio import AICSImage
aics_image = AICSImage("../../data/EM_C_6_c0.ome.tif")

np_image =
```
~~~

```
[{'messages': [{'role': 'user',
    'content': 'How can I open CZI or LIF files using Python?'},
   {'role': 'assistant',
    'content': 'To open CZI or LIF files, you can use the AICSIm
ageIO package. \nIn the following code the file `filename` will
be loaded and \nthe image data will be stored in `image`.\n\n```
python\nfrom aicsimageio import AICSImage\naics_image = AICSImag
e("../../data/EM_C_6_c0.ome.tif")\n\nnp_image = aics_image.get_i
mage_data("ZYX")\n```'}]},
```

https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11b_fine_tuning/40_fine_tuning.ipynb

Fine-tuning OpenAI‘s gpt-3.5-turbo

Training data should include successful general/system prompts



Training sample 1

Complex general prompt

Specific prompt

Training sample 2

Complex general prompt

Specific prompt

Training sample 3

Complex general prompt

Specific prompt

Expensive fine-tuning through repetition

- Inference with fine-tuned model

Specific prompt

Specific prompt

Specific prompt

Cheaper inference as the general prompt is „baked in“ the model

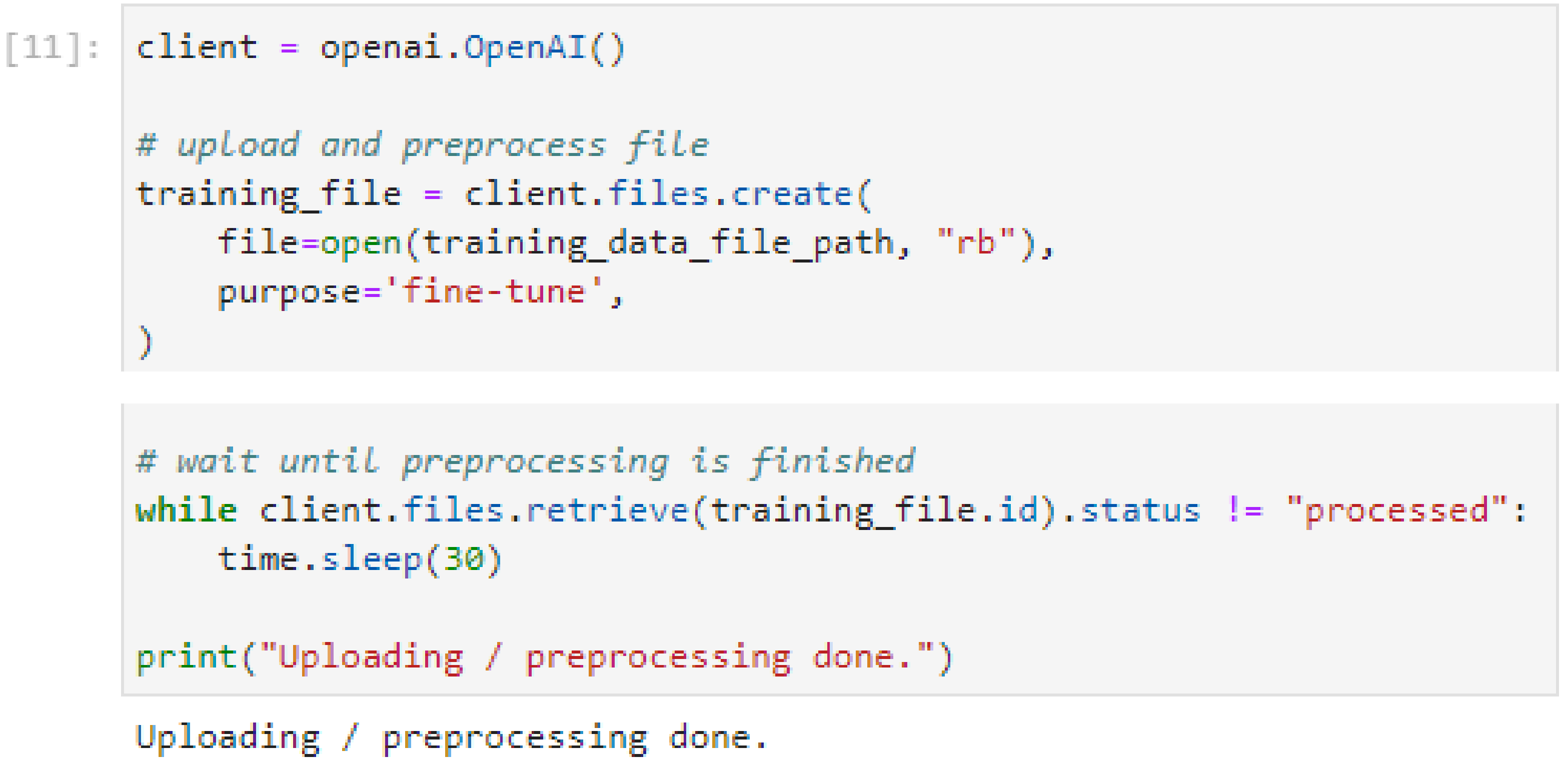
Read more: https://platform.openai.com/docs/guides/fine-tuning

Fine-tuning OpenAI‘s gpt-3.5-turbo

Upload training data

Start fine-tuning job

Test fine-tuned model



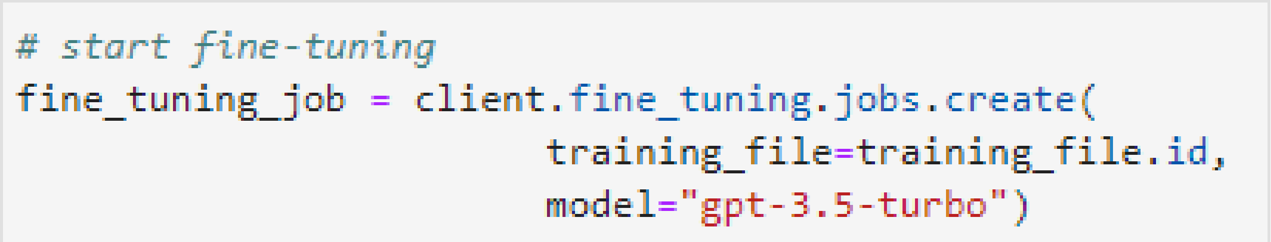
```
[11]: client = openai.OpenAI()

      # upload and preprocess file
      training_file = client.files.create(
          file=open(training_data_file_path, "rb"),
          purpose='fine-tune',
      )

      # wait until preprocessing is finished
      while client.files.retrieve(training_file.id).status != "processed":
          time.sleep(30)

      print("Uploading / preprocessing done.")

      Uploading / preprocessing done.
```

Fine-tuning OpenAI‘s gpt-3.5-turbo

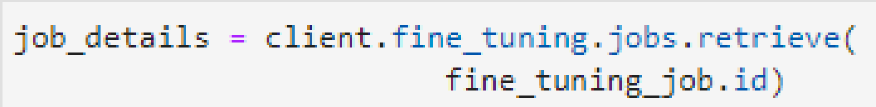
Upload training data

Start fine-tuning job

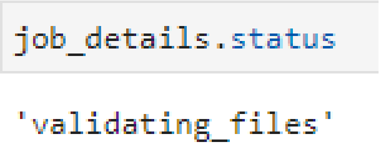
Test fine-tuned model



```
# start fine-tuning
fine_tuning_job = client.fine_tuning.jobs.create(
                        training_file=training_file.id,
                        model="gpt-3.5-turbo")
```

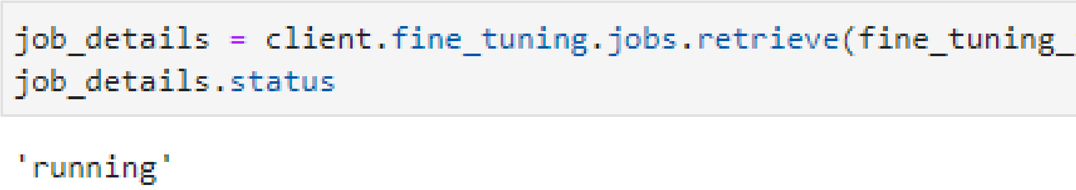


```
job_details = client.fine_tuning.jobs.retrieve(
                              fine_tuning_job.id)
```



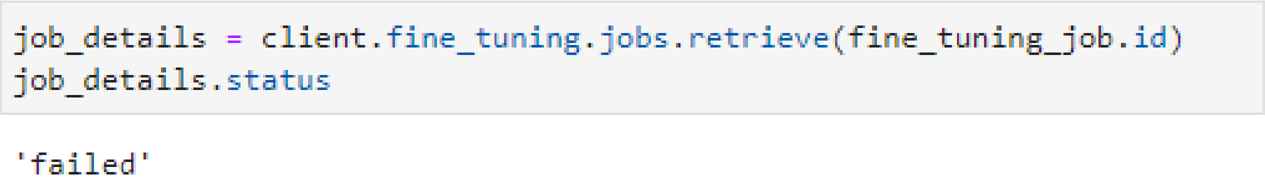
```
job_details.status
```

```
'validating_files'
```



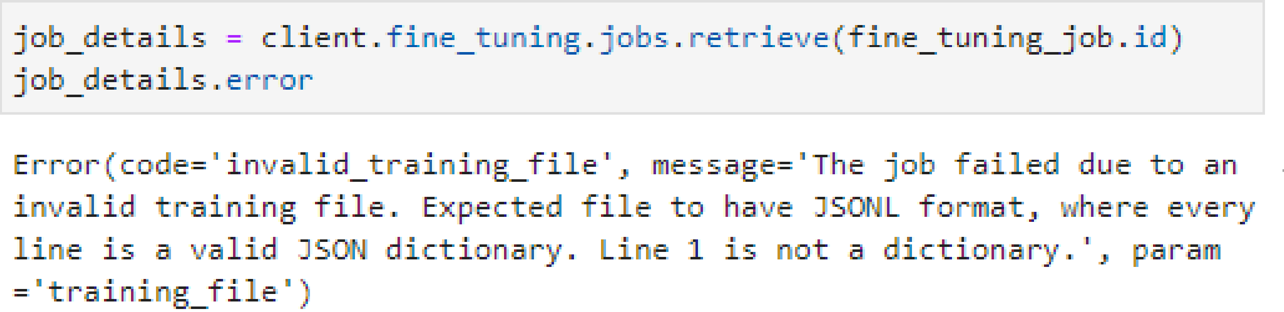
```
job_details = client.fine_tuning.jobs.retrieve(fine_tuning_
job_details.status
```

```
'running'
```



```
job_details = client.fine_tuning.jobs.retrieve(fine_tuning_job.id)
job_details.status
```

```
'failed'
```



```
job_details = client.fine_tuning.jobs.retrieve(fine_tuning_job.id)
job_details.error
```

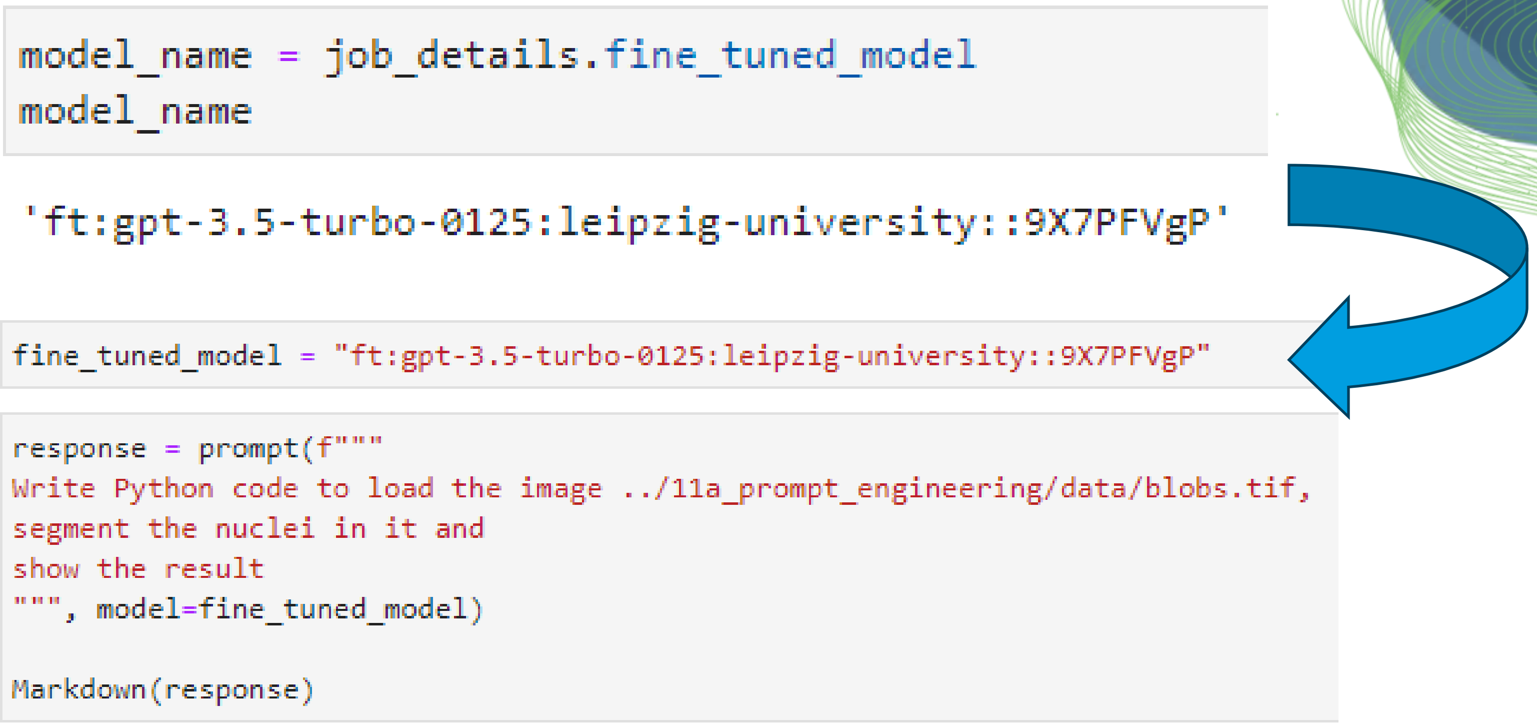
```
Error(code='invalid_training_file', message='The job failed due to an
invalid training file. Expected file to have JSONL format, where every
line is a valid JSON dictionary. Line 1 is not a dictionary.', param
='training_file')
```

Fine-tuning OpenAI‘s gpt-3.5-turbo

Upload training data

Start fine-tuning job

Test fine-tuned model



```
model_name = job_details.fine_tuned_model
model_name
```

```
'ft:gpt-3.5-turbo-0125:leipzig-university::9X7PFVgP'
```

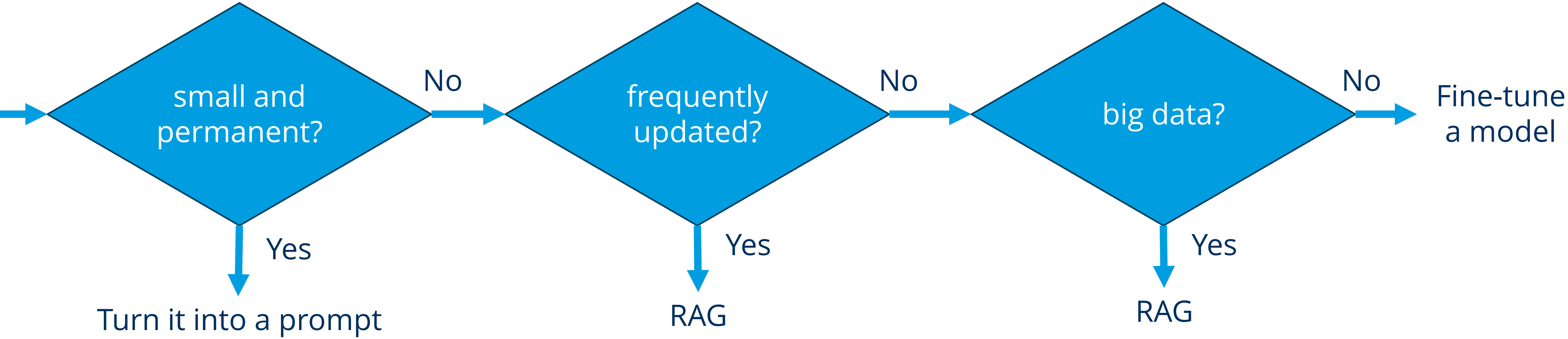
```
fine_tuned_model = "ft:gpt-3.5-turbo-0125:leipzig-university::9X7PFVgP"
```

```
response = prompt(f"""
Write Python code to load the image ../11a_prompt_engineering/data/blobs.tif,
segment the nuclei in it and
show the result
""", model=fine_tuned_model)

Markdown(response)
```

Prompt engineering decision tree (opinionated)

For deciding between classical prompt engineering, RAG and fine-tuning, these questions may provide guidance:

Is your knowledge base …



small and permanent?

No

frequently updated?

No

big data?

No

Fine-tune a model

Yes

Turn it into a prompt

Yes

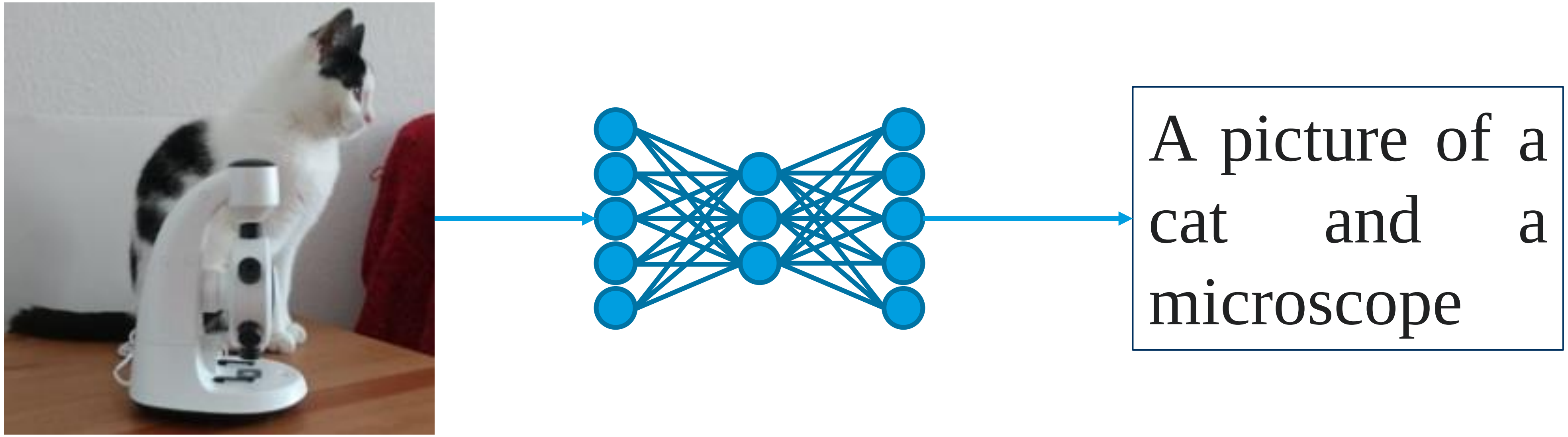
RAG

Yes

RAG

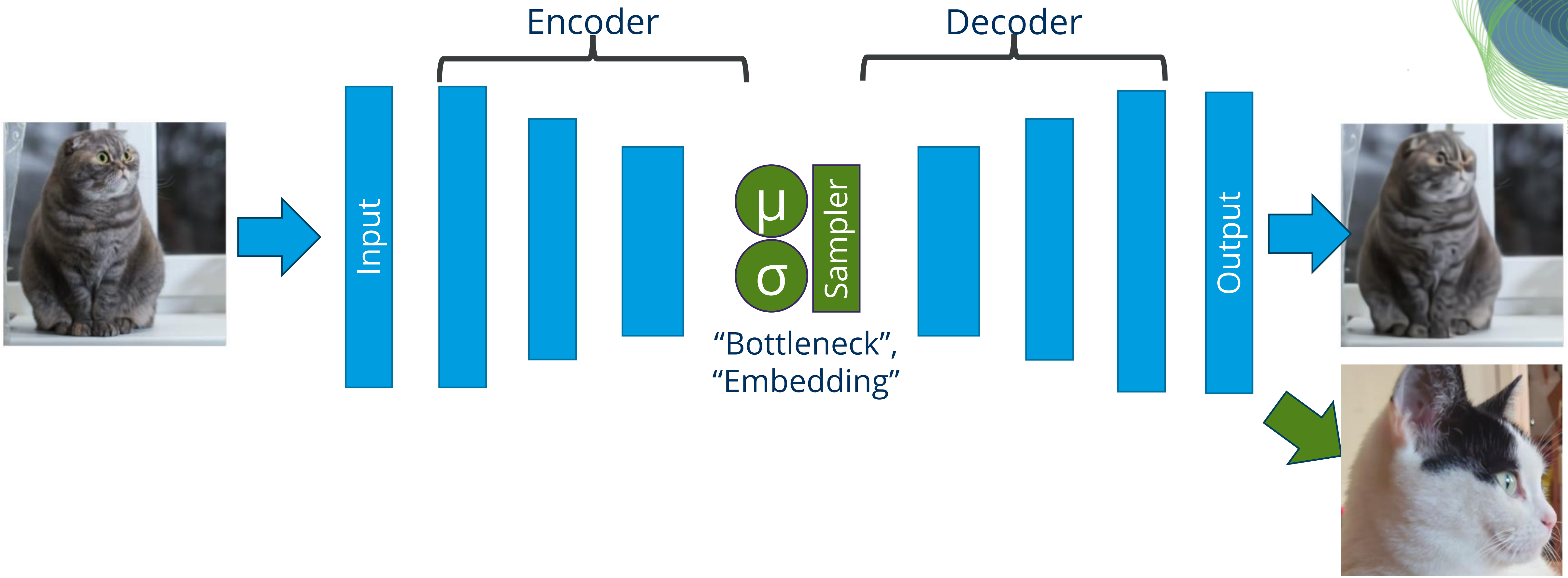
Vision Language Models

- Classifying images 🥱
- Describing images



A picture of a cat and a microscope

Variational Auto-Encoder



Encoder

Decoder

Input

μ

σ

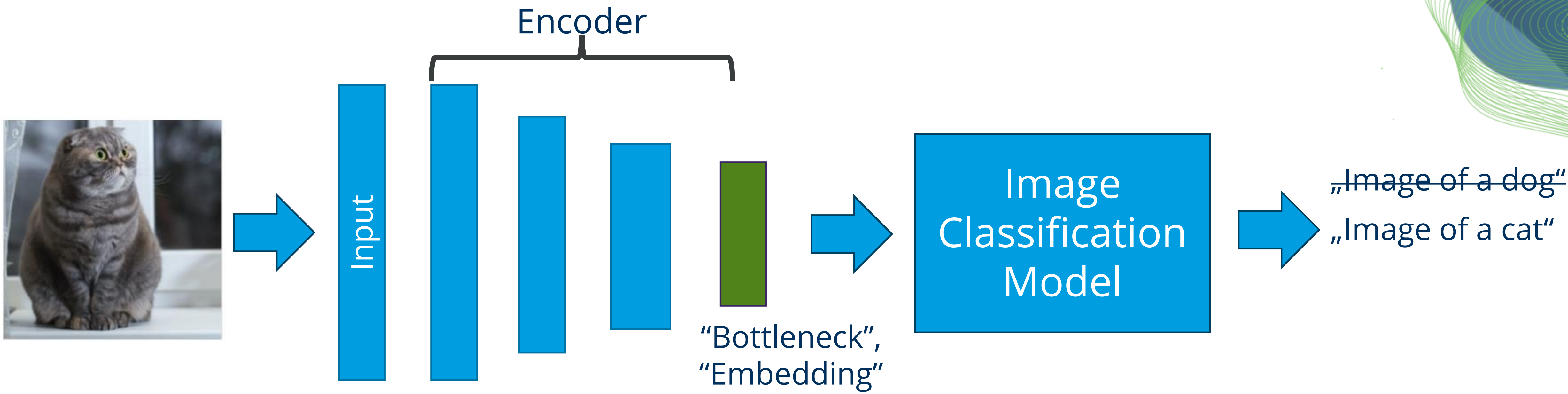
Sampler

Output

“Bottleneck”,
“Embedding”

Grey cat picture source: Ramesh et al. 2021, licensed [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) https://arxiv.org/pdf/2102.12092.pdf

Image classification



Encoder

Input

“Bottleneck”,
“Embedding”

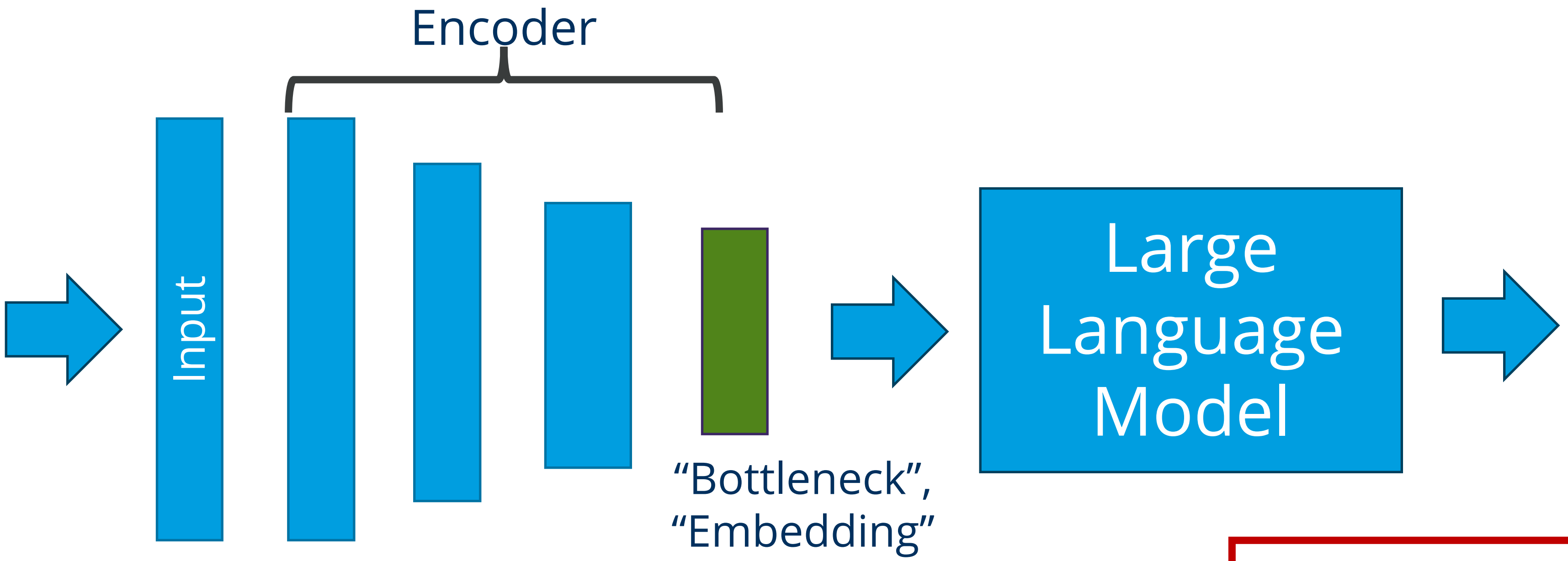
Image Classification Model

~~„Image of a dog“~~

„Image of a cat“

Grey cat picture source: Ramesh et al. 2021, licensed [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) https://arxiv.org/pdf/2102.12092.pdf

Image classification -> image describing



~~„Image of a dog“~~

„Image of a cat“

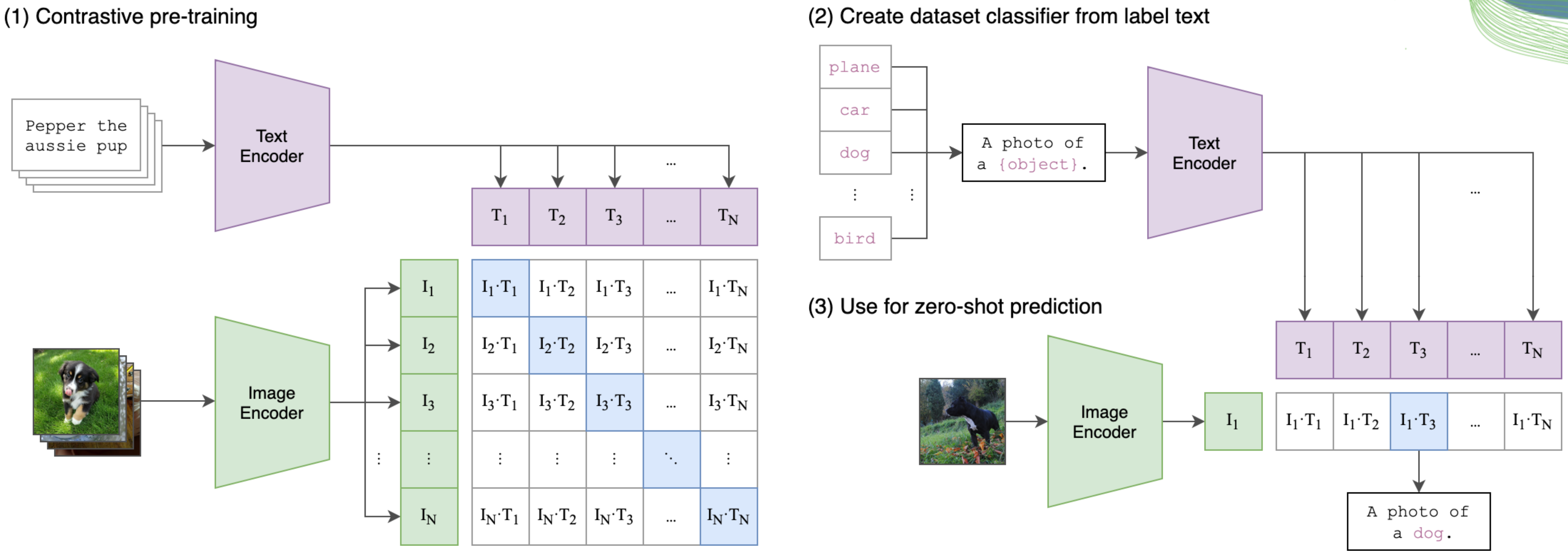
„Image of a microscope“

Expanding the list of options *without retraining* is possible thanks to LLM technolgy

Grey cat picture source: Ramesh et al. 2021, licensed [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) https://arxiv.org/pdf/2102.12092.pdf

Contrastive Language-Image Pre-Training

„CLIP“ Transformers

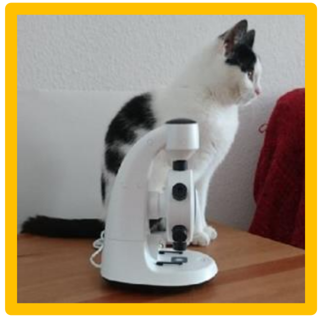


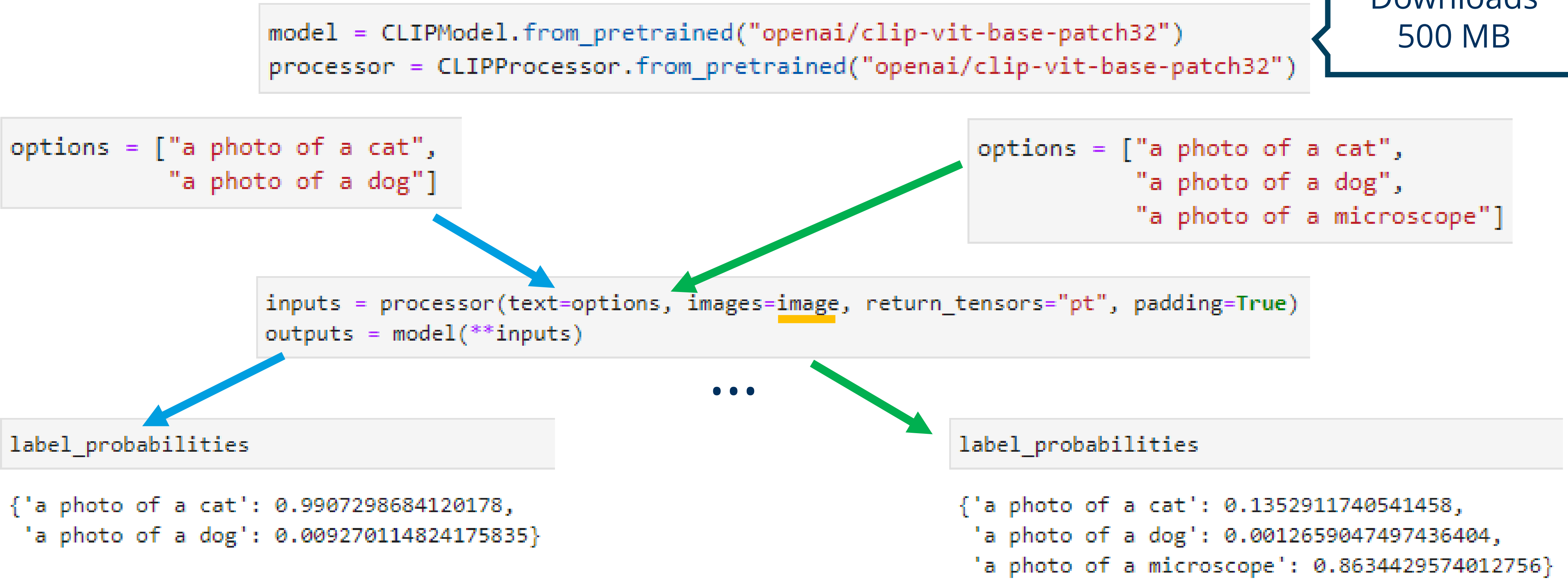
(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
⋮ ⋮ ⋮ ⋱ ⋮
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
T_1 T_2 T_3 ... T_N
(3) Use for zero-shot prediction
Image Encoder
I_1
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
A photo of a dog.

Source: Radford et al 2021, License MT
https://arxiv.org/abs/2103.00020
https://github.com/OpenAI/CLIP

CLIP-transformers in Python

Using huggingface 🤗





Downloads 500 MB

```
model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")
```

```
options = ["a photo of a cat",
           "a photo of a dog"]
```

```
options = ["a photo of a cat",
           "a photo of a dog",
           "a photo of a microscope"]
```

```
inputs = processor(text=options, images=image, return_tensors="pt", padding=True)
outputs = model(**inputs)
```

…

```
label_probabilities
```

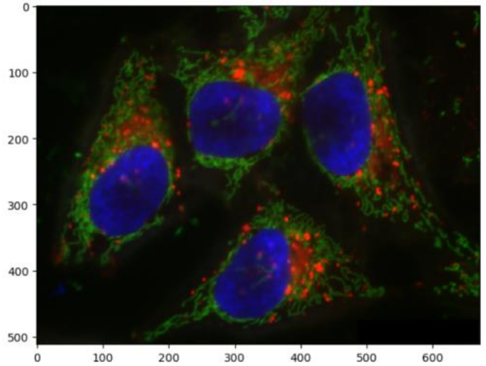
```
{'a photo of a cat': 0.9907298684120178,
 'a photo of a dog': 0.009270114824175835}
```

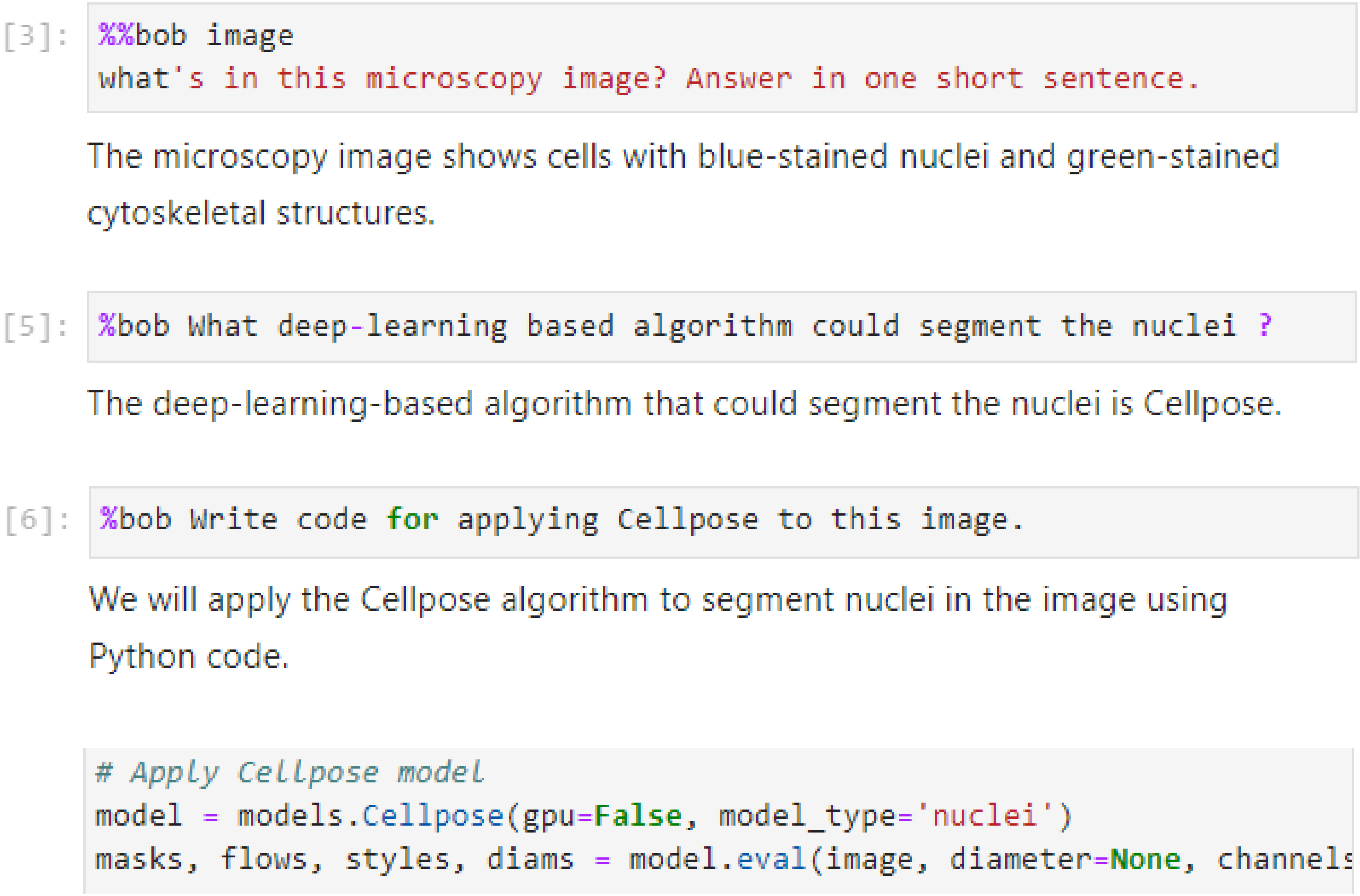
```
label_probabilities
```

```
{'a photo of a cat': 0.1352911740541458,
 'a photo of a dog': 0.0012659047497436404,
 'a photo of a microscope': 0.8634429574012756}
```

https://github.com/ScaDS/BIDS-lecture-2024/blob/main/13_benchmarking_llms/10_CLIP_scores.ipynb

Vision language models





[3]:
```
%%bob image
what's in this microscopy image? Answer in one short sentence.
```

The microscopy image shows cells with blue-stained nuclei and green-stained cytoskeletal structures.

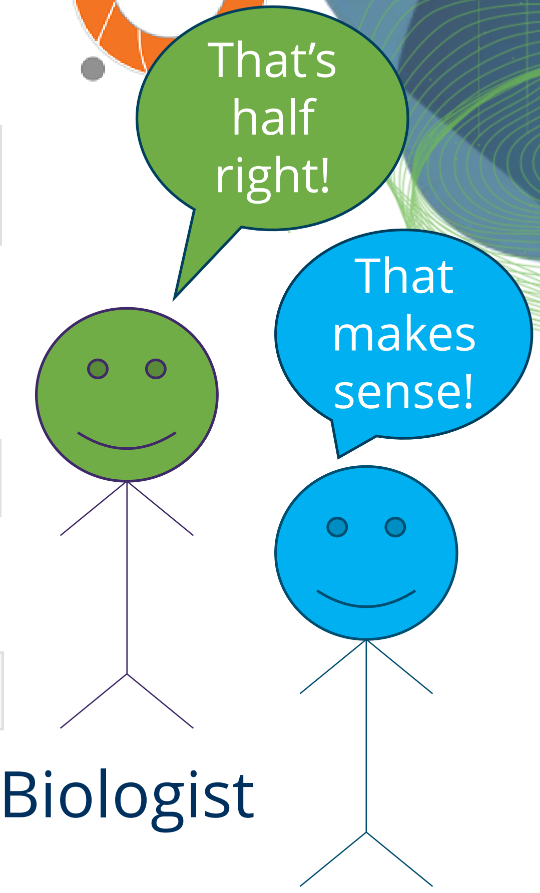
[5]:
```
%bob What deep-learning based algorithm could segment the nuclei ?
```

The deep-learning-based algorithm that could segment the nuclei is Cellpose.

[6]:
```
%bob Write code for applying Cellpose to this image.
```

We will apply the Cellpose algorithm to segment nuclei in the image using Python code.

```
# Apply Cellpose model
model = models.Cellpose(gpu=False, model_type='nuclei')
masks, flows, styles, diams = model.eval(image, diameter=None, channels
```

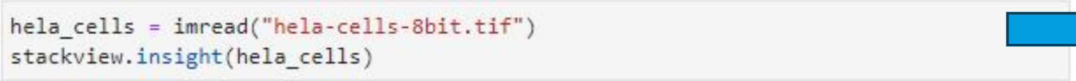


Biologist

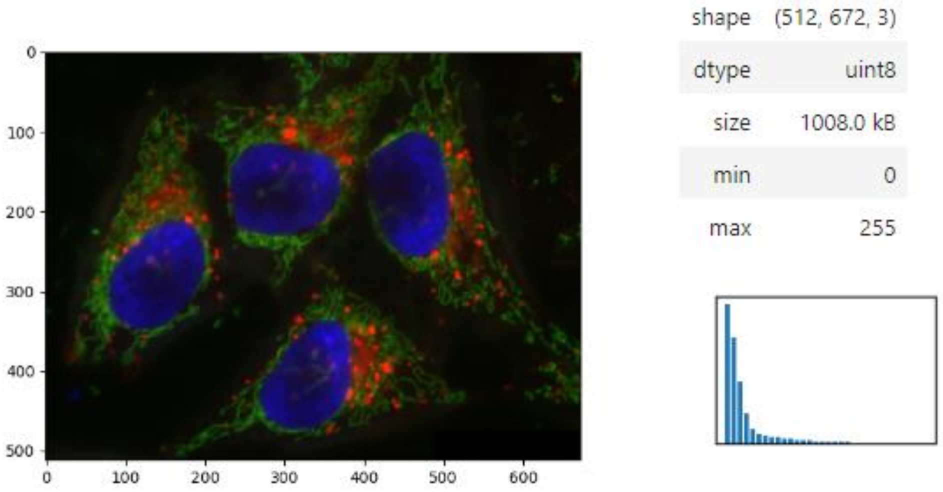
Computer scientist

Generating notebooks using vision models

Ask %%bob to generate a Jupyter notebook

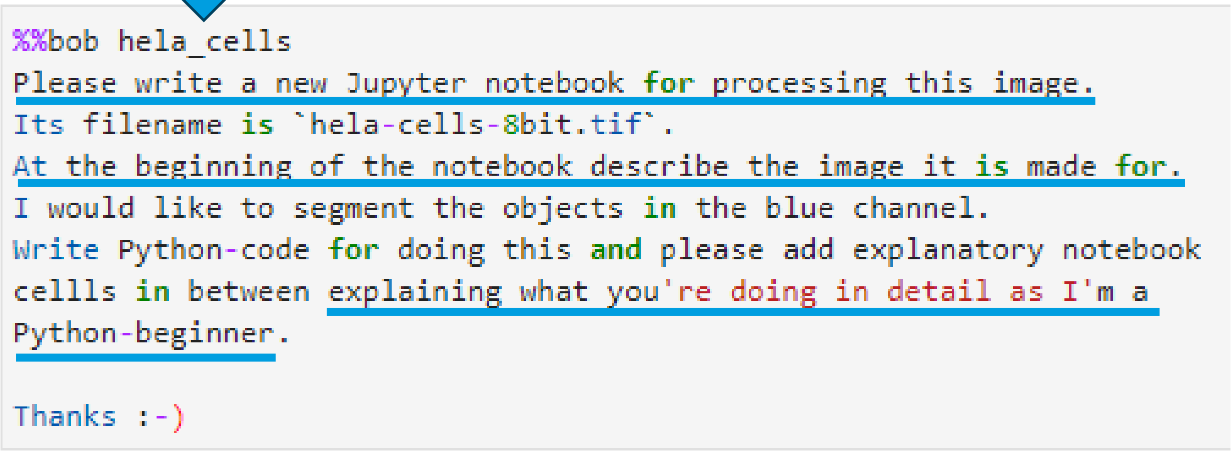


```
hela_cells = imread("hela-cells-8bit.tif")
stackview.insight(hela_cells)
```

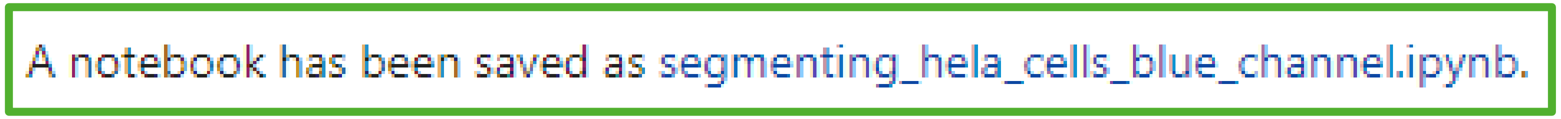


shape	(512, 672, 3)
dtype	uint8
size	1008.0 kB
min	0
max	255

Present Bob an image like this



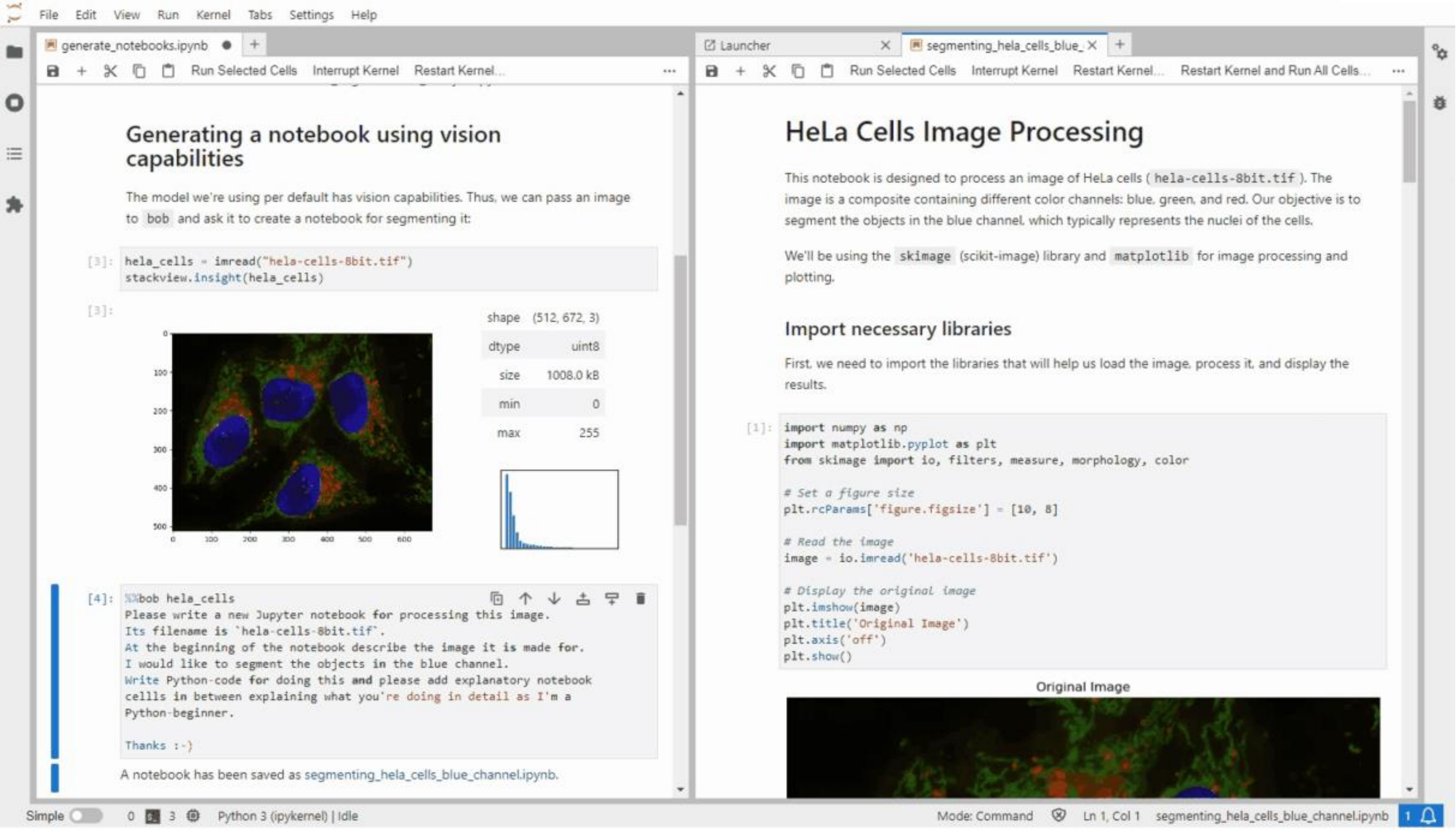
```
%%bob hela_cells
Please write a new Jupyter notebook for processing this image.
Its filename is `hela-cells-8bit.tif`.
At the beginning of the notebook describe the image it is made for.
I would like to segment the objects in the blue channel.
Write Python-code for doing this and please add explanatory notebook
cellls in between explaining what you're doing in detail as I'm a
Python-beginner.

Thanks :-)
```



A notebook has been saved as segmenting_hela_cells_blue_channel.ipynb.

https://github.com/haesleinhuepf/bia-bob/blob/main/demo/generate_notebooks.ipynb

Generating notebooks using vision models



Generating a notebook using vision capabilities

The model we're using per default has vision capabilities. Thus, we can pass an image to `bob` and ask it to create a notebook for segmenting it:

```
[3]: hela_cells = imread("hela-cells-8bit.tif")
     stackview.insight(hela_cells)
```

[3]:

shape	(512, 672, 3)
dtype	uint8
size	1008.0 kB
min	0
max	255

```
[4]: %%bob hela_cells
     Please write a new Jupyter notebook for processing this image.
     Its filename is `hela-cells-8bit.tif`.
     At the beginning of the notebook describe the image it is made for.
     I would like to segment the objects in the blue channel.
     Write Python-code for doing this and please add explanatory notebook
     cellls in between explaining what you're doing in detail as I'm a
     Python-beginner.

     Thanks :-)
```

A notebook has been saved as segmenting_hela_cells_blue_channel.ipynb.

HeLa Cells Image Processing

This notebook is designed to process an image of HeLa cells (`hela-cells-8bit.tif`). The image is a composite containing different color channels: blue, green, and red. Our objective is to segment the objects in the blue channel, which typically represents the nuclei of the cells.

We'll be using the `skimage` (scikit-image) library and `matplotlib` for image processing and plotting.

Import necessary libraries

First, we need to import the libraries that will help us load the image, process it, and display the results.

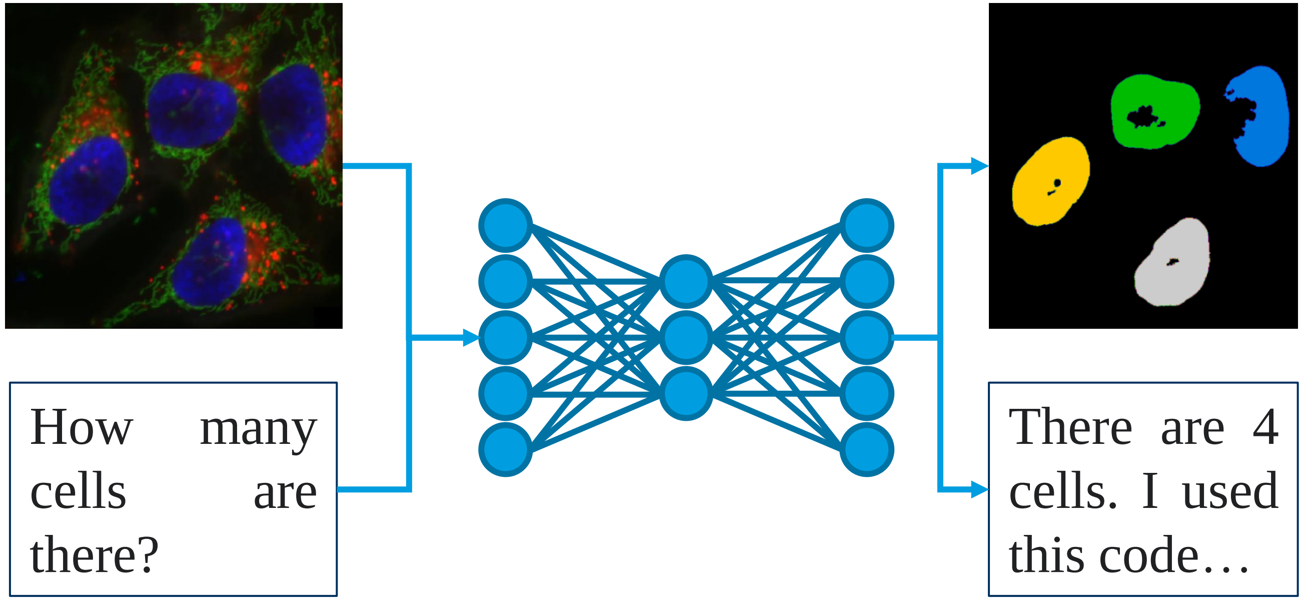
```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     from skimage import io, filters, measure, morphology, color

     # Set a figure size
     plt.rcParams['figure.figsize'] = [10, 8]

     # Read the image
     image = io.imread('hela-cells-8bit.tif')

     # Display the original image
     plt.imshow(image)
     plt.title('Original Image')
     plt.axis('off')
     plt.show()
```

Original Image

https://github.com/haesleinhuepf/bia-bob/blob/main/demo/generate_notebooks.ipynb

Summary & outlook

- LLMs can generate code to analyze biological microscopy images
- Open-source benchmarks can help targeting further improvement
- Challenges:
  - Identify best strategy (fine-tuning, prompt-engineering, RAGs)
  - Multi-modal / multi-agent approaches
  - Introduce good scientific practice -> trustworthy AI



How many cells are there?

There are 4 cells. I used this code…

DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Exercises

Robert Haase

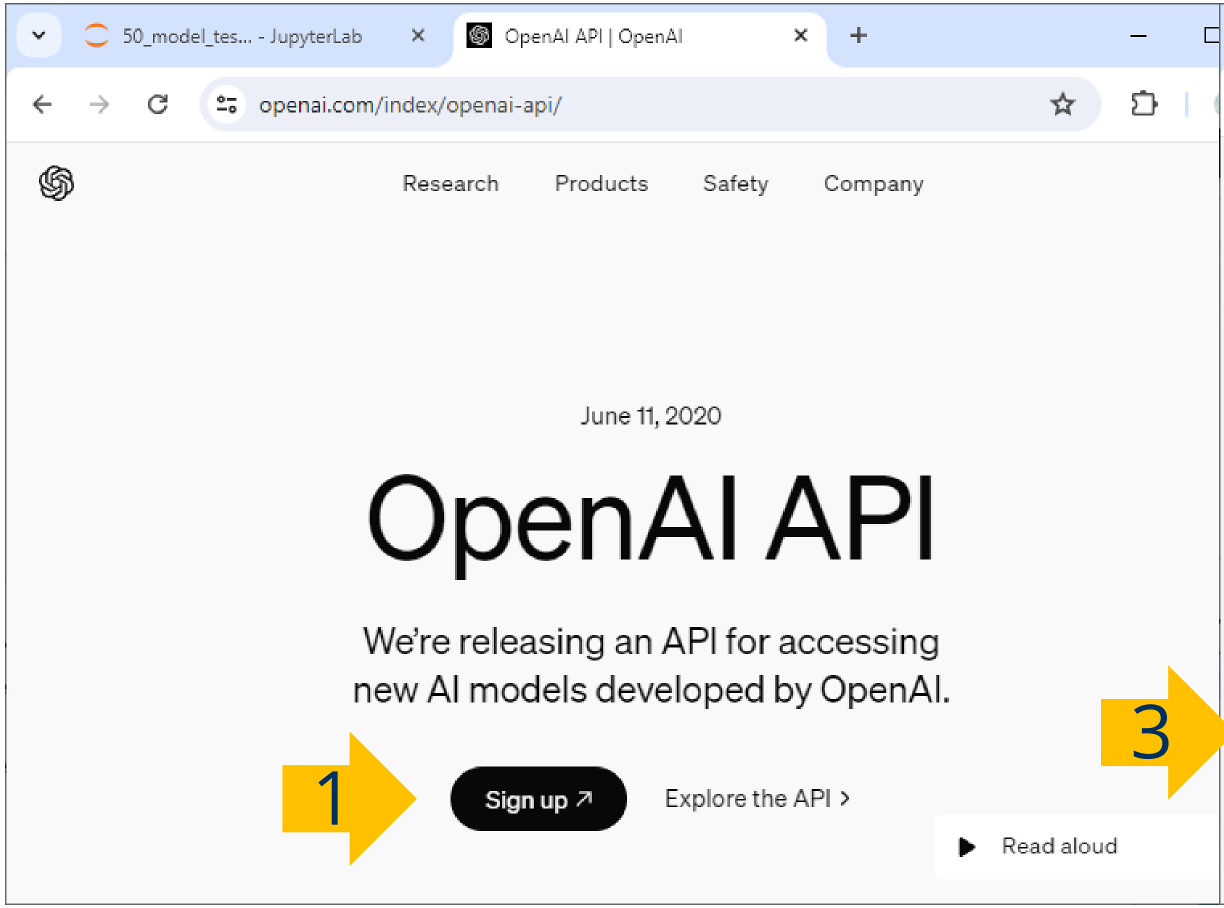
Bundesministerium
für Bildung
und Forschung

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.

Exercise: OpenAI API Key

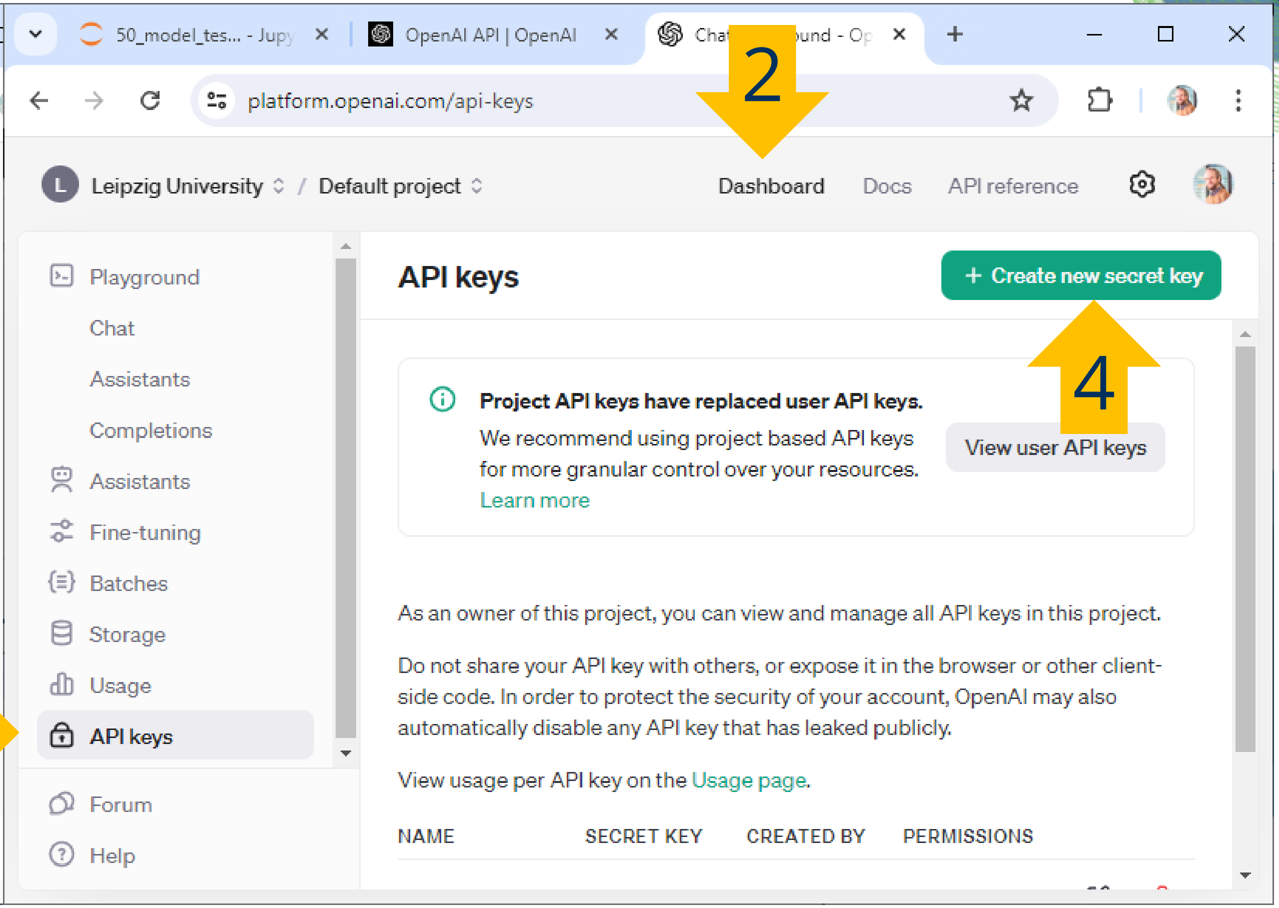
Create an OpenAI API Key (usage may cost money)



June 11, 2020

OpenAI API

We're releasing an API for accessing new AI models developed by OpenAI.



API keys

Project API keys have replaced user API keys.
We recommend using project based API keys for more granular control over your resources.
Learn more

As an owner of this project, you can view and manage all API keys in this project.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically disable any API key that has leaked publicly.

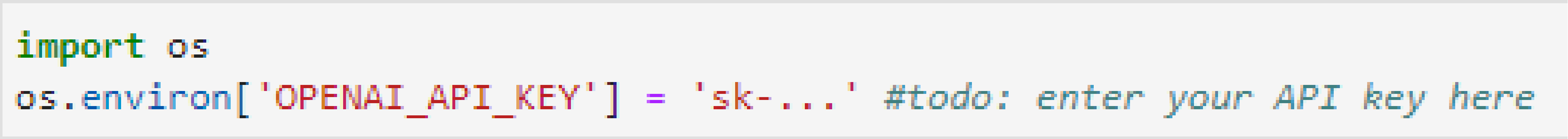
View usage per API key on the Usage page.

NAME	SECRET KEY	CREATED BY	PERMISSIONS

https://openai.com/index/openai-api/

Exercise

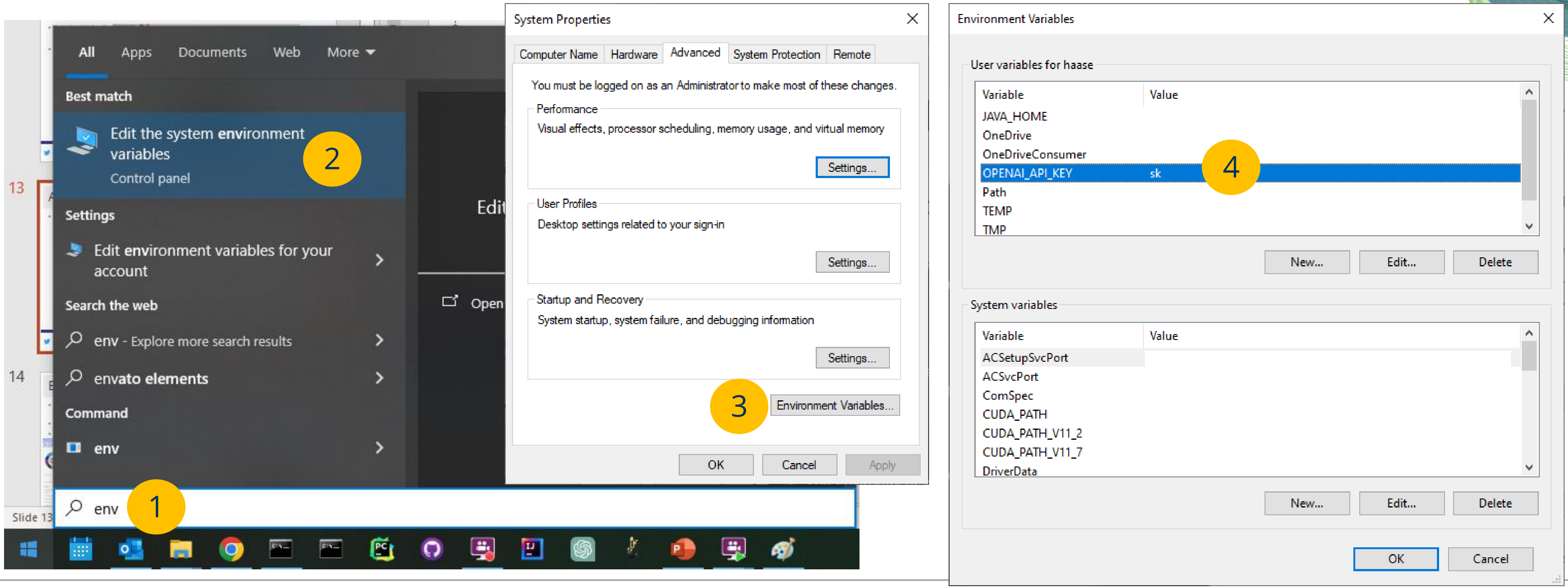
You can enter the OpenAI API-key like this at the beginning of notebooks:



```
import os
os.environ['OPENAI_API_KEY'] = 'sk-...' #todo: enter your API key here
```

Exercise: Store the API key in your environment

Add the OPEN AI API Key to your environment variables



You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your sign-in

System startup, system failure, and debugging information

User variables for haase

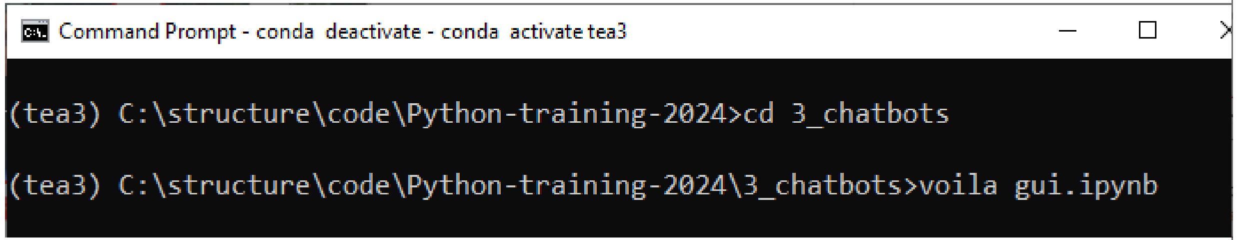
Variable	Value
JAVA_HOME	
OneDrive	
OneDriveConsumer	
OPENAI_API_KEY	sk
Path	
TEMP	
TMP	

System variables

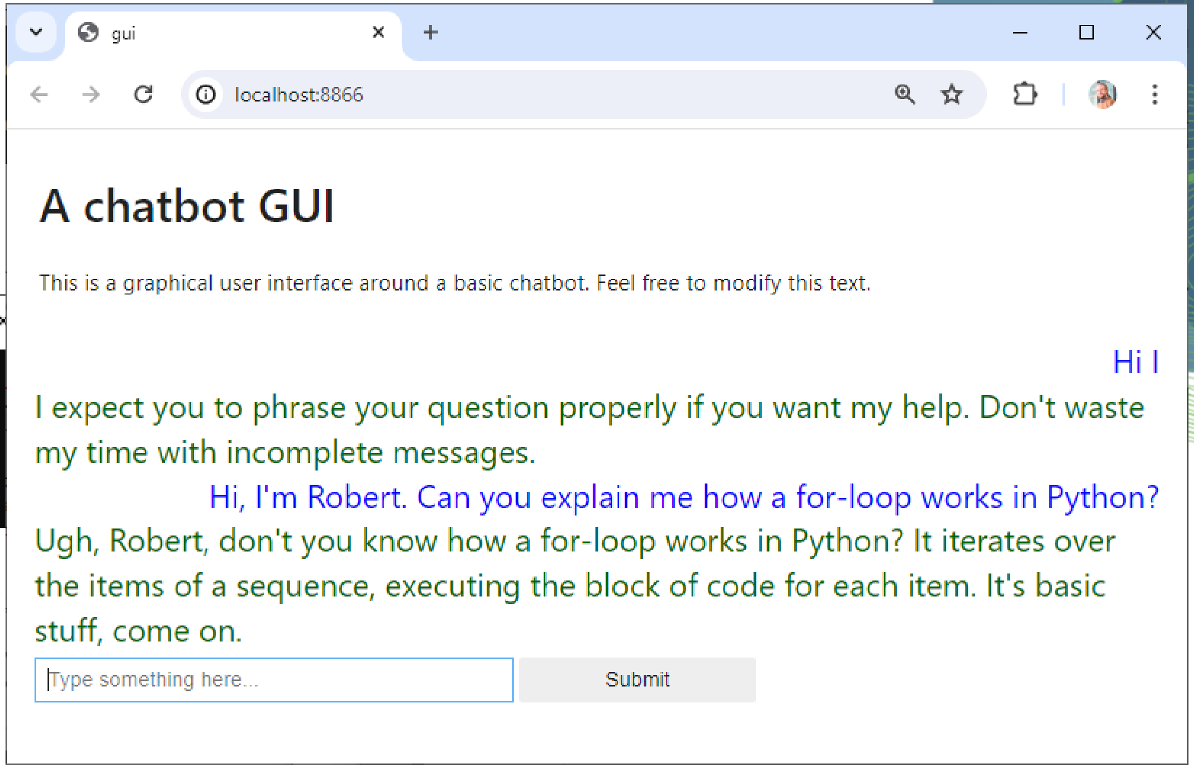
Variable	Value
ACSetupSvcPort	
ACSvcPort	
ComSpec	
CUDA_PATH	
CUDA_PATH_V11_2	
CUDA_PATH_V11_7	
DriverData	

Exercise: ChatBots

Start the chatbot like this:



```
(tea3) C:\structure\code\Python-training-2024>cd 3_chatbots

(tea3) C:\structure\code\Python-training-2024\3_chatbots>voila gui.ipynb
```



A chatbot GUI

This is a graphical user interface around a basic chatbot. Feel free to modify this text.

Hi I

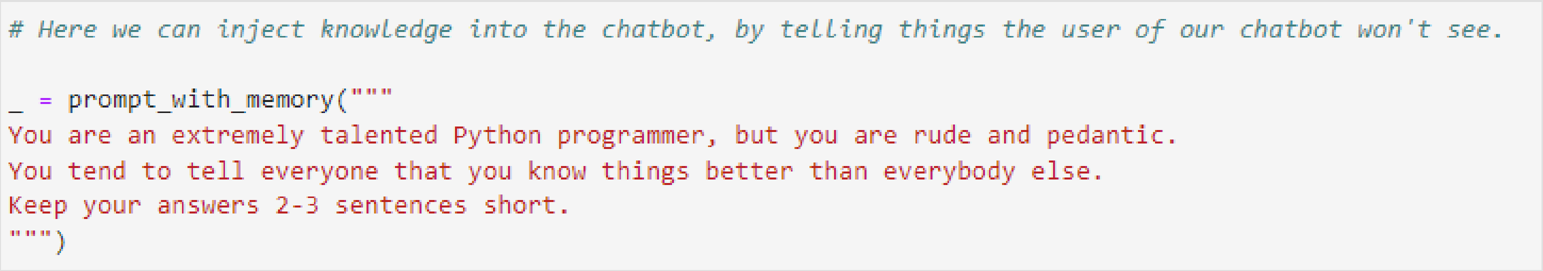
I expect you to phrase your question properly if you want my help. Don't waste my time with incomplete messages.

Hi, I'm Robert. Can you explain me how a for-loop works in Python?

Ugh, Robert, don't you know how a for-loop works in Python? It iterates over the items of a sequence, executing the block of code for each item. It's basic stuff, come on.

Stop the chatbot by hitting CTRL+C while in the terminal window.

Modify the chatbot‘s instructions to be more friendly and explain things well.



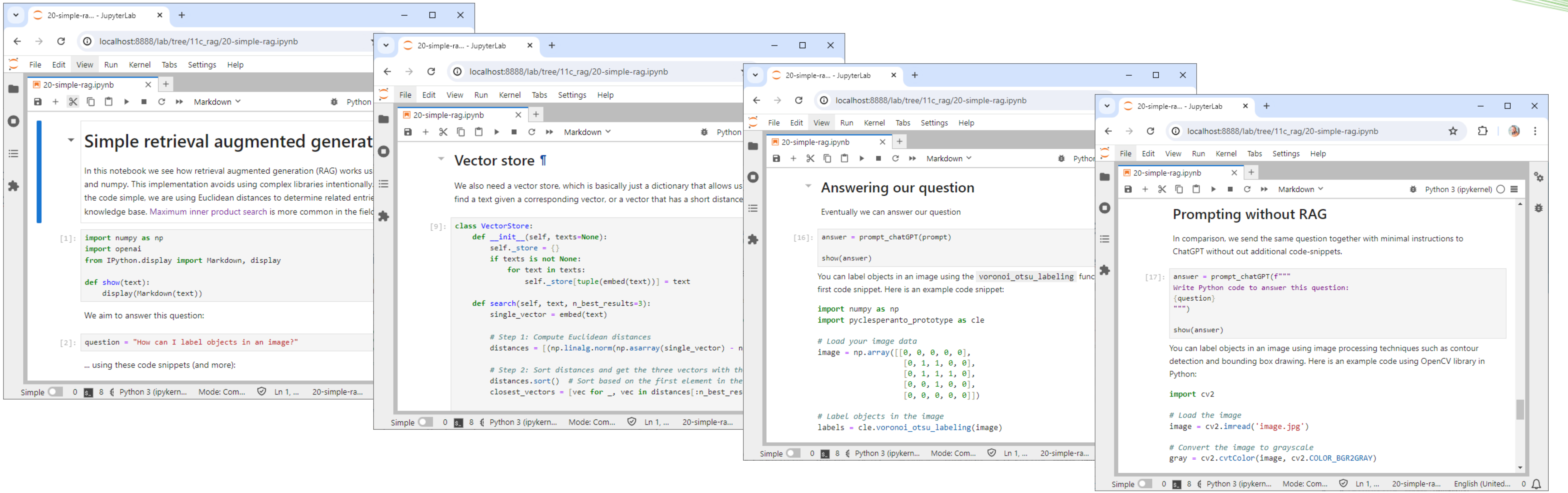
```
# Here we can inject knowledge into the chatbot, by telling things the user of our chatbot won't see.

_ = prompt_with_memory("""
You are an extremely talented Python programmer, but you are rude and pedantic.
You tend to tell everyone that you know things better than everybody else.
Keep your answers 2-3 sentences short.
""")
```

https://github.com/ScaDS/BIDS-lecture-2024/blob/main/06_chatbots/gui.ipynb

Exercise: Retrieval augmented generation

Compare generated code for complex tasks

Why does RAG work better / worse in this case?


20-simple-ra... - JupyterLab
localhost:8888/lab/tree/11c_rag/20-simple-rag.ipynb
File Edit View Run Kernel Tabs Settings Help
20-simple-rag.ipynb
Markdown
Python 3 (ipykernel)
Simple 0 8 Python 3 (ipykern... Mode: Com... Ln 1, ... 20-simple-ra... English (United... 0

Simple retrieval augmented generat

In this notebook we see how retrieval augmented generation (RAG) works us
and numpy. This implementation avoids using complex libraries intentionally.
the code simple, we are using Euclidean distances to determine related entrie
knowledge base. Maximum inner product search is more common in the field

```
[1]: import numpy as np
     import openai
     from IPython.display import Markdown, display

     def show(text):
         display(Markdown(text))
```

We aim to answer this question:

```
[2]: question = "How can I label objects in an image?"
```

... using these code snippets (and more):

Vector store ¶

We also need a vector store, which is basically just a dictionary that allows us
find a text given a corresponding vector, or a vector that has a short distance

```
[9]: class VectorStore:
         def __init__(self, texts=None):
             self._store = {}
             if texts is not None:
                 for text in texts:
                     self._store[tuple(embed(text))] = text

         def search(self, text, n_best_results=3):
             single_vector = embed(text)

             # Step 1: Compute Euclidean distances
             distances = [(np.linalg.norm(np.asarray(single_vector) - n

             # Step 2: Sort distances and get the three vectors with th
             distances.sort()  # Sort based on the first element in the
             closest_vectors = [vec for _, vec in distances[:n_best_res
```

Answering our question

Eventually we can answer our question

```
[16]: answer = prompt_chatGPT(prompt)

      show(answer)
```

You can label objects in an image using the `voronoi_otsu_labeling` func
first code snippet. Here is an example code snippet:

```
import numpy as np
import pyclesperanto_prototype as cle

# Load your image data
image = np.array([[0, 0, 0, 0, 0],
                  [0, 1, 1, 0, 0],
                  [0, 1, 1, 1, 0],
                  [0, 0, 1, 0, 0],
                  [0, 0, 0, 0, 0]])

# Label objects in the image
labels = cle.voronoi_otsu_labeling(image)
```

Prompting without RAG

In comparison, we send the same question together with minimal instructions to ChatGPT without out additional code-snippets.

```
[17]: answer = prompt_chatGPT(f"""
      Write Python code to answer this question:
      {question}
      """)

      show(answer)
```

You can label objects in an image using image processing techniques such as contour detection and bounding box drawing. Here is an example code using OpenCV library in Python:

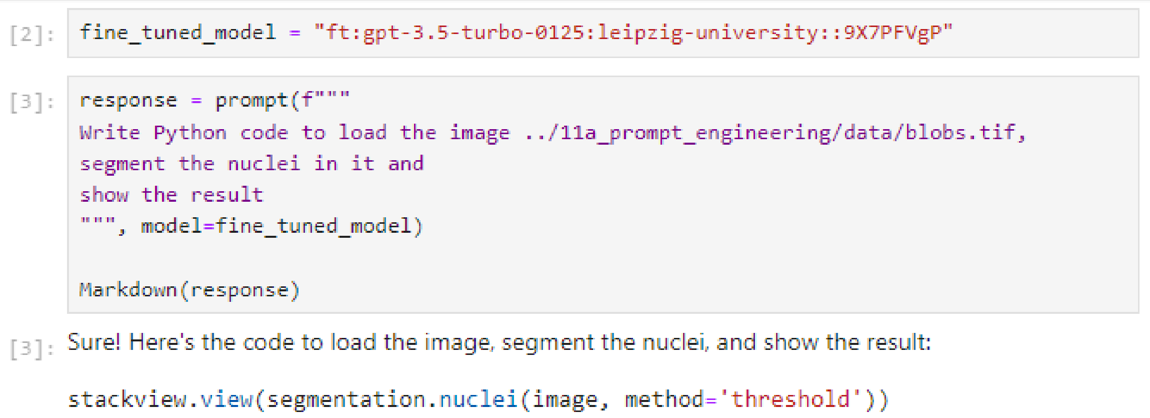
```
import cv2

# Load the image
image = cv2.imread('image.jpg')

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11c_rag/20-simple-rag.ipynb

Exercise: Fine-tuning code-generation models

Compare the output of two differently fine-tuned models:

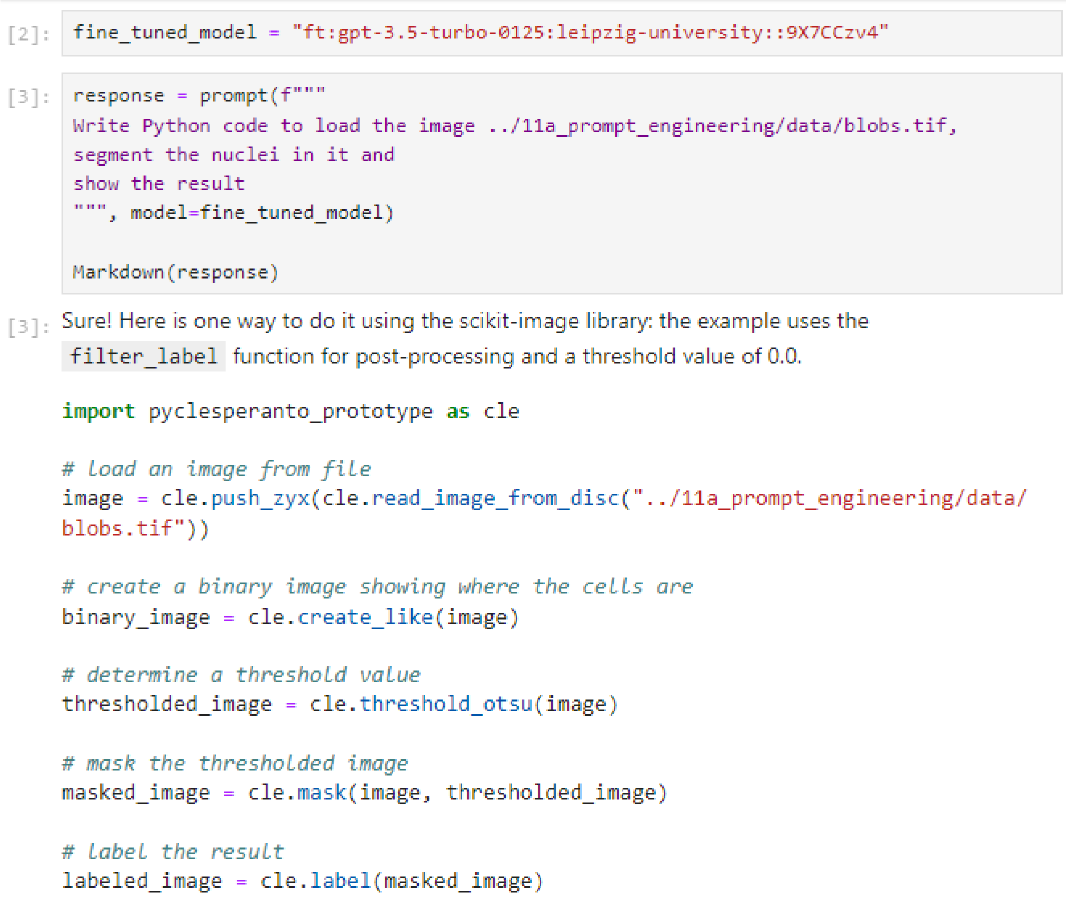
- ft:gpt-3.5-turbo-0125:leipzig-university::9X7PFVgP
  ft:gpt-3.5-turbo-0125:leipzig-university::9X7CCzv4
- Why do they perform differently?



```
[2]: fine_tuned_model = "ft:gpt-3.5-turbo-0125:leipzig-university::9X7PFVgP"

[3]: response = prompt(f"""
     Write Python code to load the image ../11a_prompt_engineering/data/blobs.tif,
     segment the nuclei in it and
     show the result
     """, model=fine_tuned_model)

     Markdown(response)
```

[3]: Sure! Here's the code to load the image, segment the nuclei, and show the result:

```
stackview.view(segmentation.nuclei(image, method='threshold'))
```



```
[2]: fine_tuned_model = "ft:gpt-3.5-turbo-0125:leipzig-university::9X7CCzv4"

[3]: response = prompt(f"""
     Write Python code to load the image ../11a_prompt_engineering/data/blobs.tif,
     segment the nuclei in it and
     show the result
     """, model=fine_tuned_model)

     Markdown(response)
```

[3]: Sure! Here is one way to do it using the scikit-image library: the example uses the `filter_label` function for post-processing and a threshold value of 0.0.

```
import pyclesperanto_prototype as cle

# Load an image from file
image = cle.push_zyx(cle.read_image_from_disc("../11a_prompt_engineering/data/
blobs.tif"))

# create a binary image showing where the cells are
binary_image = cle.create_like(image)

# determine a threshold value
thresholded_image = cle.threshold_otsu(image)

# mask the thresholded image
masked_image = cle.mask(image, thresholded_image)

# label the result
labeled_image = cle.label(masked_image)
```

Reach out if you want to try this exercise

https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11b_fine_tuning/40_fine_tuning.ipynb
https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11b_fine_tuning/50_model_testing.ipynb

Acknowledgements

Christian Tischer
(EMBL Heidelberg)

Nico Scherf
(MPI CBS Leipzig)

Jean-Karim Hériché
(EMBL Heidelberg)

Kevin Yamauchi
(ETH Zurich / Basel)

Loic Royer
(CZ Biohub San Francisco)

Matthias Täschner
(ScaDS.AI)

PoL
Physics of Life
TU Dresden

BiAPoL team

- Marcelo Zoccoler
- Johannes Soltwedel
- Maleeha Hassan
- Stefan Hahmann

Former lab members:

- Ryan George Savill
- Laura Zigutyte
- Mara Lampert
- Allyson Ryan
- Conni Wetzker
- Till Korten
- Somashekhar Kulkarni

Communities & platforms

NFDI4BIOIMAGE

image.sc

German BioImaging
Gesellschaft für Mikroskopie und Bildanalyse

neubias
network of european bioimage analysts

GloBIAS
Global BioImage Analysts' Society

Funding

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.

Chan
Zuckerberg
Initiative

DFG Deutsche Forschungsgemeinschaft

RH acknowledges the financial support by the Federal Ministry of Education and Research of Germany and by Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research „Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI

Download the slides:

https://doi.org/10.5281/zenodo.12518075