

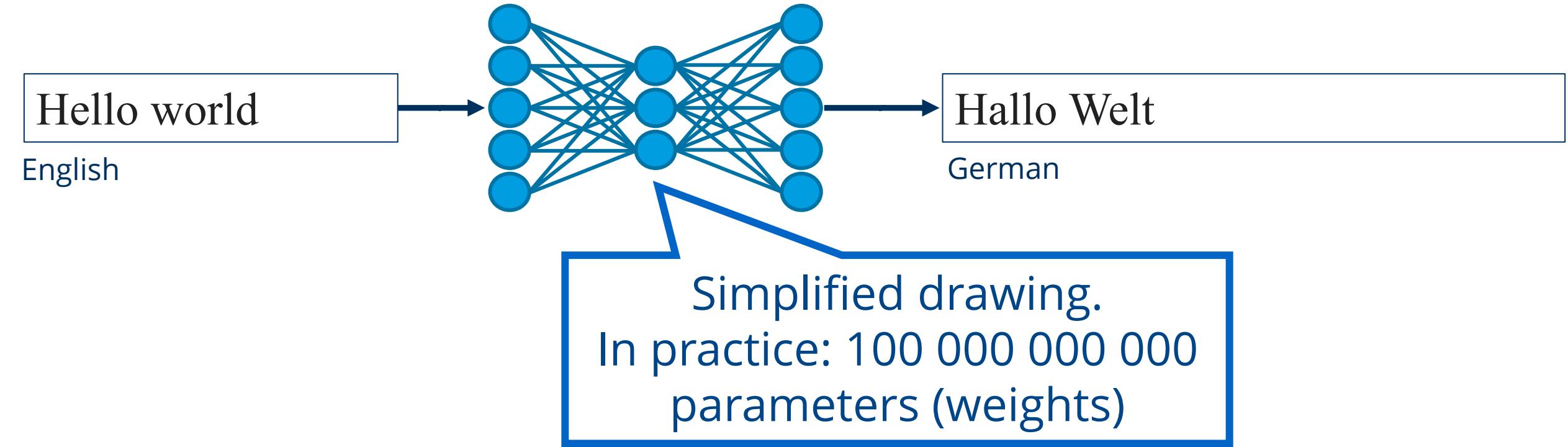
# Using Large Language Models for Data Analysis

## Robert Haase

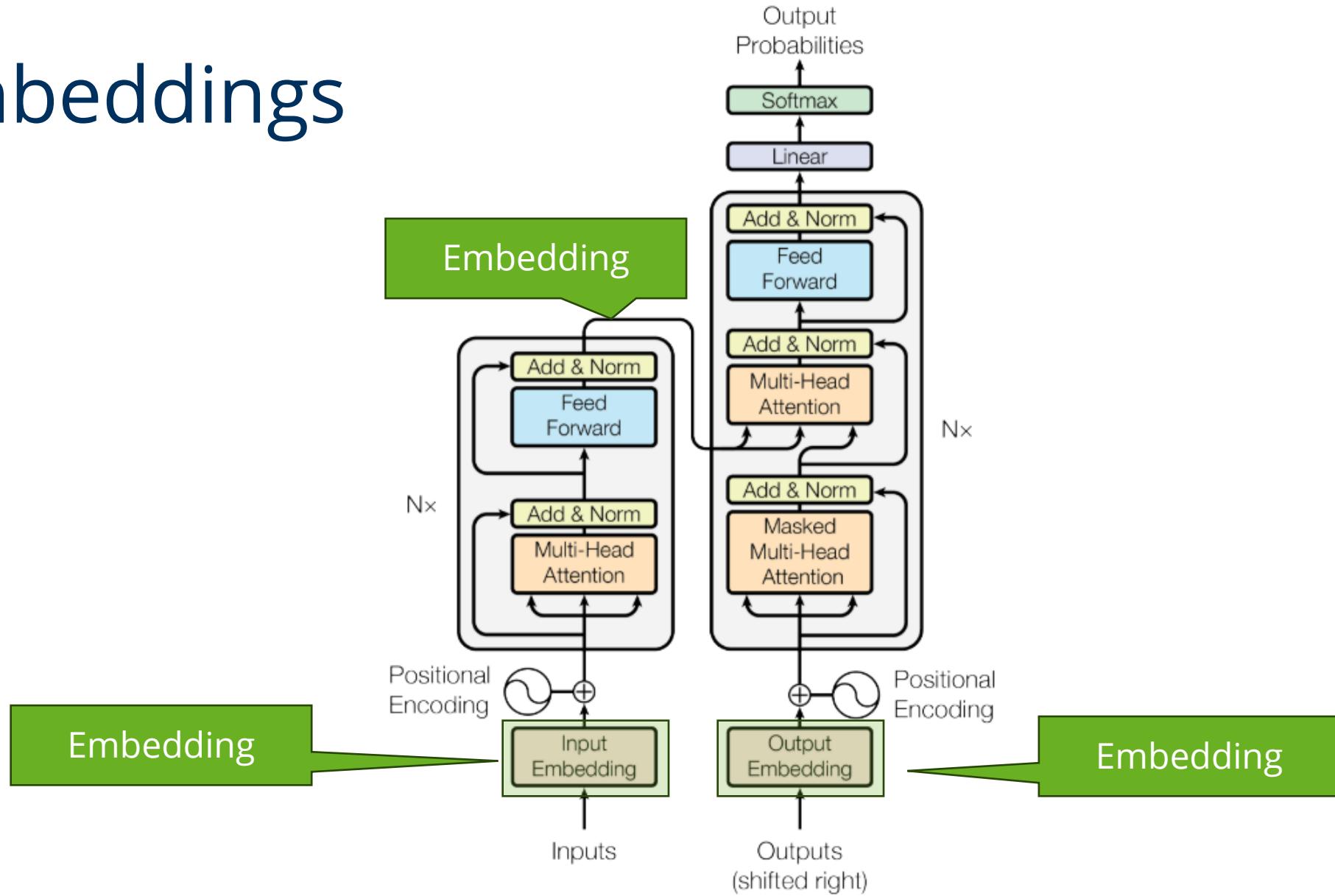
These slides can be reused under the terms of the [CC-BY4.0](#) license.

# Large Language Models (LLMs)

Text-to-text, translation



# Embeddings



# Embeddings

N-dimensional vector space representing... a word, sentence, text, ...

```
def embed(text, model="Qwen/Qwen3-Embedding-4B"):
    import os
    from openai import OpenAI
    client = OpenAI(base_url="https://llm.scads.ai/v1",
                    api_key=os.environ.get('SCADSAI_API_KEY'))

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

Connecting to  
the server

Submitting a  
prompt

NEVER submit  
confidential,  
secret or private  
information!

# Embeddings

N-dimensional vector space representing... a word, sentence, text, ...

```
def embed(text, model="Qwen/Qwen3-Embedding-4B"):
    import os
    from openai import OpenAI
    client = OpenAI(base_url="https://llm.scads.ai/v1",
                    api_key=os.environ.get('SCADSAI_API_KEY'))

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

```
v = embed("hello world")
```

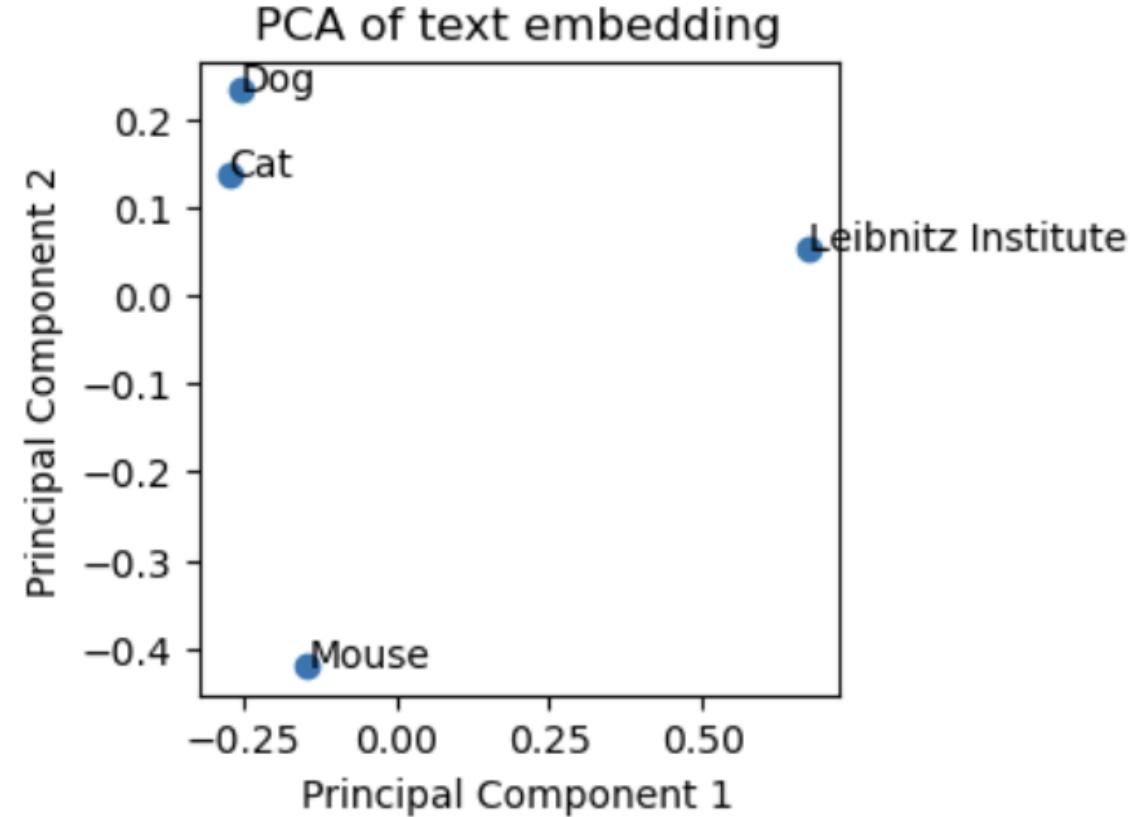
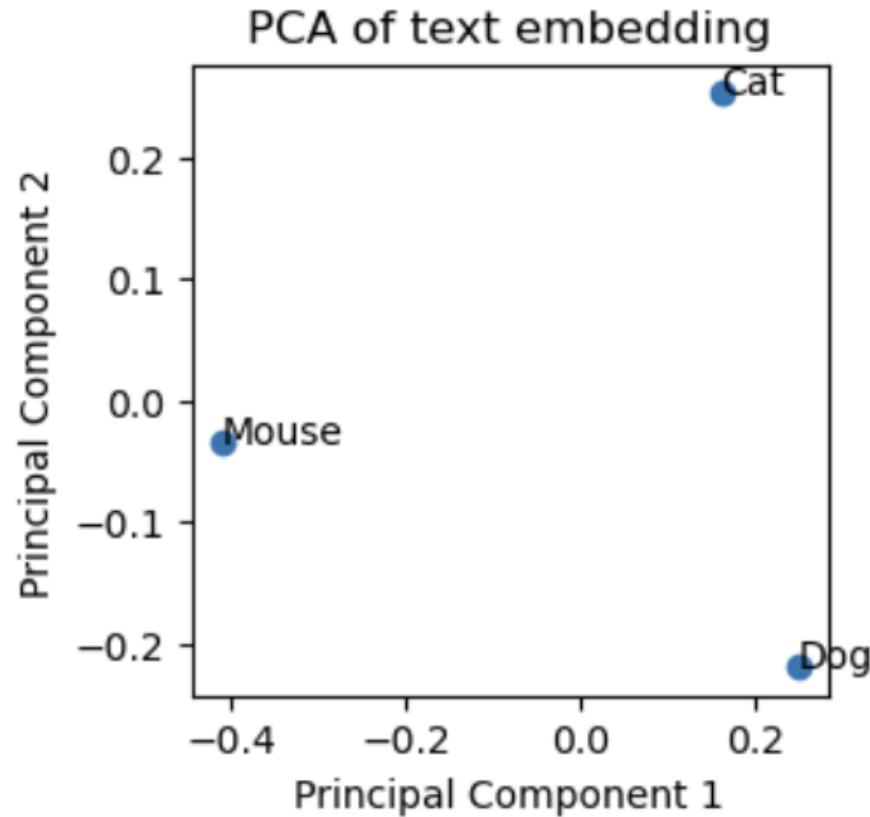
```
len(v)
```

```
v[:5]
```

```
[0.0001030213970807381,
 -0.02199709229171276,
 0.011862305924296379,
 0.03385939821600914,
 0.0005146571202203631]
```

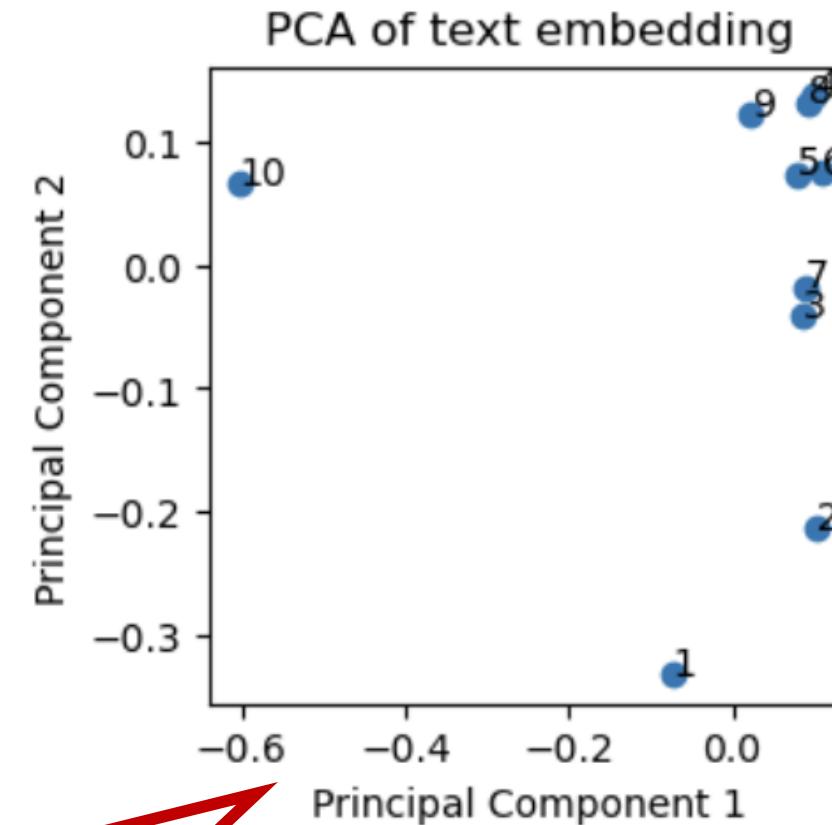
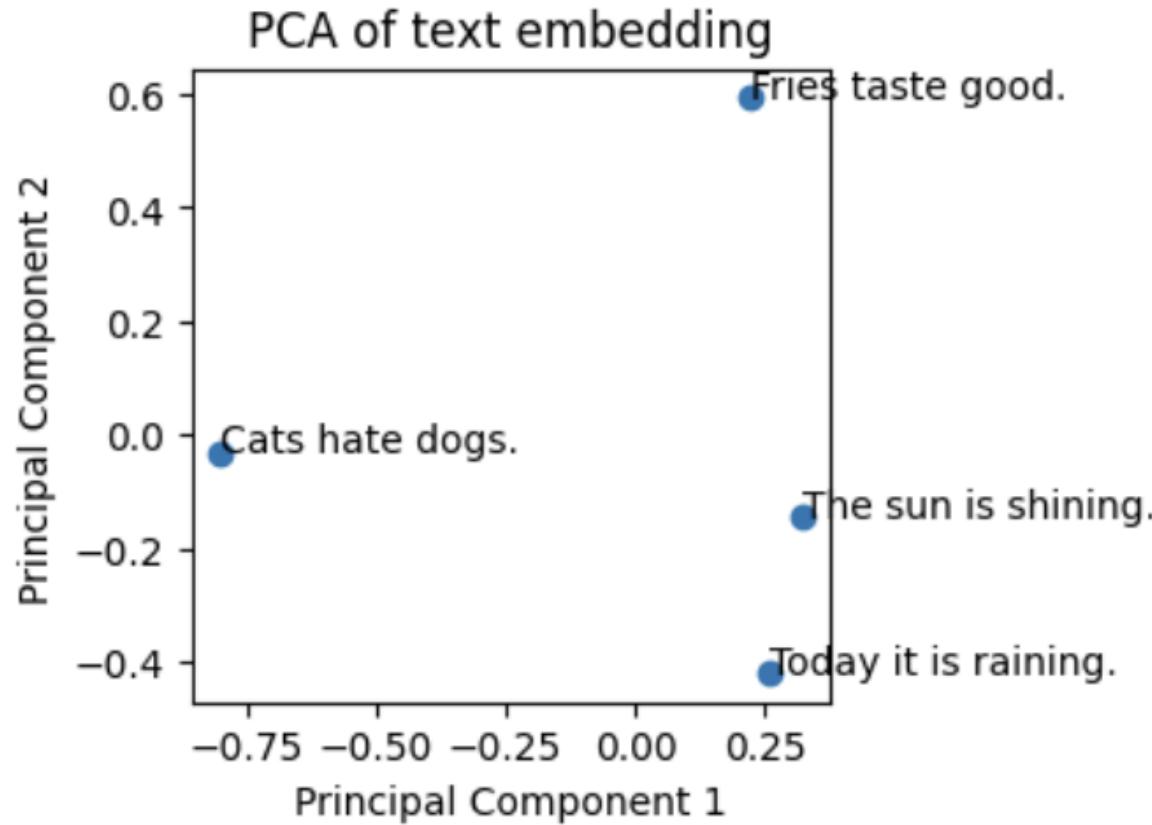
# Embeddings + Dimensionality reduction

... simply because we cannot display 2560 dimensions on a 2D screen.



# Embeddings + Dimensionality reduction

... simply because we cannot display 2560 dimensions on a 2D screen.



LLMs are not made  
/ meant for this

# Embeddings

Just for fun, I took all Leibnitz MMS institutes and generated PhD thesis topics that could be done at these institutes. Then, I generated embeddings for them.

	TSNE0	TSNE1	UMAP0	UMAP1	embedding	name	research_field	selection	topic
0	2.548442	8.969955	-1.020463	9.660064	[0.019196026027202606, 0.010897933505475521, -...]	Taylor Reed	FIZ-KA - Leibniz-Institut für Informationsinfrastruktur	1	Digital Archives, Embodied Knowledge, and the ...
1	-12.006737	-13.473205	1.419240	6.479001	[0.016622617840766907, -0.009818249382078648, -...]	Riley Jain	HKI - Leibniz-Institut für Naturstoff-Forschung und Innovation	1	Microbial Secondary Metabolites and Narrative...
2	-10.385350	12.543368	4.321546	4.795953	[-0.016480615362524986, 0.014093692414462566, -...]	Taylor Adams	IÖR - Leibniz-Institut für ökologische Raumentwicklung	1	Spatial Imaginaries of Ecological Transition: ...
3	8.713301	-10.254994	0.332513	10.496520	[0.008821303024888039, 0.005257884040474892, -...]	Devon Thomas	IWM - Leibniz-Institut für Wissensmedien, Tübingen	1	Algorithmic Storytelling and the Evolution of ...
4	-12.952323	3.004944	2.245961	3.266387	[-0.02530881017446518, 0.004650192800909281, -...]	Alex Lee	MfN - Museum für Naturkunde - Leibniz-Institut für Naturkunde	1	The Literary Ecology of Scientific Illustration

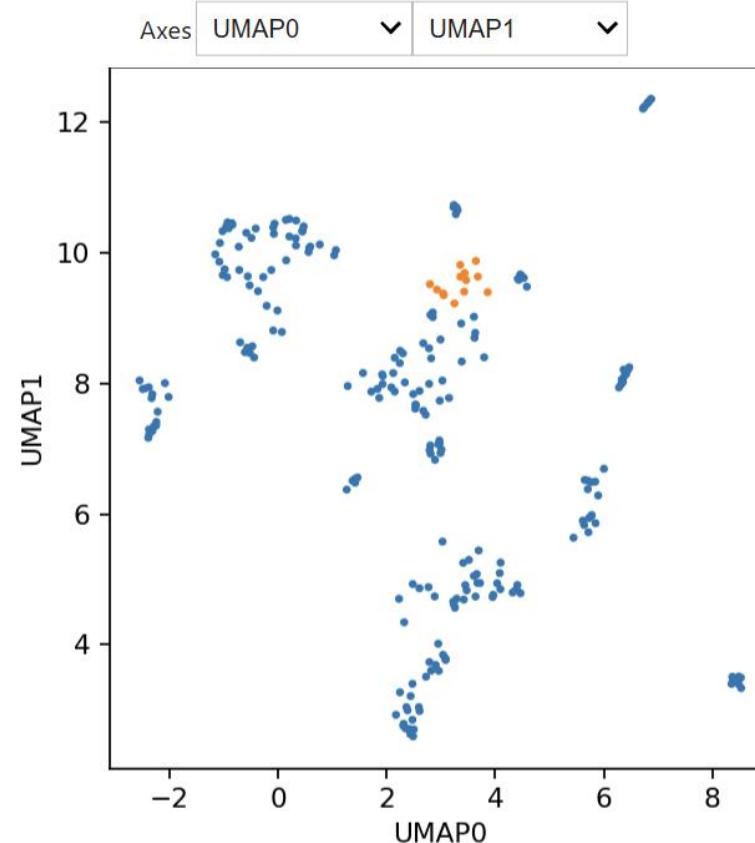
# Quiz guessing Institutes



Show solution

Reset

FBH - Ferdinand-Braun-Institut, Leibniz-Institut für Höchstfrequenztechnik, Berlin  
INP - Leibniz-Institut für Plasmaforschung und Technologie e.V. Greifswald  
IPHT - Leibniz-Institut für Photonische Technologien, Jena  
MBI - Max-Born-Institut für Nichtlineare Optik und Kurzzeitspektroskopie, Berlin



# Using LLMs in Python

```
def prompt_scadsai_llm(message:str, model="meta-llama/Llama-4-Scout-17B-16E-Instruct"):
    """A prompt helper function that sends a message to ScaDS.AI LLM server at
    ZIH TU Dresden and returns only the text response.
    """

    import os

    # convert message in the right format if necessary
    if isinstance(message, str):
        message = [{"role": "user", "content": message}]

    # setup connection to the LLM
    client = openai.OpenAI(base_url="https://llm.scads.ai/v1",
                           api_key=os.environ.get('SCADSAI_API_KEY'))
    response = client.chat.completions.create(
        model=model,
        messages=message
    )

    # extract answer
    return response.choices[0].message.content
```

Data wrangling

Connecting to  
the server

Submitting a  
prompt

NEVER submit  
confidential,  
secret or private  
information!

# Using LLMs in Python

```
prompt_scadsai_llm("Hi!")
```

'How can I assist you today?'

```
prompt_scadsai_llm("What is the meaning of life?")
```

"A question that has puzzled philosophers, theologians, and everyday people for centuries! There is no one definitive answer, as the meaning of life can vary greatly depending on individual perspectives, cultural backgrounds, and personal experiences. However, here are some insights and perspectives that might help:

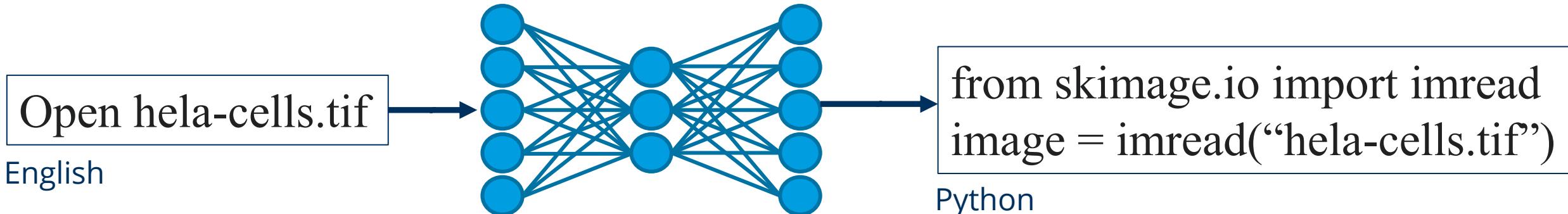
- \*\*Philosophical perspectives:\*\*
  - 1. \*\*Hedonism\*\*: The pursuit of happiness and pleasure is the primary goal of life. (Epicurus)
  - 2. \*\*Eudaimonism\*\*: Living a virtuous life, cultivating one's character, and achieving human flourishing is the key to a fulfilling existence. (Aristotle)
  - 3. \*\*Existentialism\*\*: Life has no inherent meaning; individuals must create their own purpose and meaning through their choices and actions. (Jean-Paul Sartre)
  - 4. \*\*Stoicism\*\*: The focus is on living in accordance with reason, accepting the present moment, and cultivating inner strength and wisdom. (Epicurus)
- \*\*Religious and spiritual perspectives:\*\*
  - 1. \*\*Christianity\*\*: The meaning of life is to love and serve God, and to follow Jesus Christ's teachings to achieve salvation and eternal life.
  - 2. \*\*Buddhism\*\*: The goal is to attain enlightenment (Nirvana) by following the Eightfold Path, letting go of attachments, and cultivating mindfulness and compassion.
  - 3. \*\*Islam\*\*: The purpose of life is to submit to Allah's will, follow the Quran and the Prophet Muhammad's teachings, and strive for spiritual growth and self-improvement.
- \*\*Scientific and evolutionary perspectives:\*\*
  - 1. \*\*Biological imperative\*\*: The primary drive of life is to survive, reproduce, and pass on genetic information to future generations.
  - 2. \*\*Evolutionary purpose\*\*: The meaning of life is to contribute to the continuation and evolution of the species, ensuring the survival and adaptation of humanity.
- \*\*Personal and subjective perspectives:\*\*
  - 1. \*\*Self-actualization\*\*: The meaning of life is to realize one's full potential, pursue one's passions, and become the best version of oneself. (Abraham Maslow)
  - 2. \*\*Relationships and connections\*\*: Life's purpose is to form meaningful relationships, build community, and experience love and connection with others.
  - 3. \*\*Personal growth and learning\*\*: The journey of life is about continuous learning, self-improvement, and exploration.

Ultimately, the meaning of life is a highly subjective and personal question. What gives life meaning for one person may not be the same for another. It's up to each individual to reflect on their values, beliefs, and experiences to find their own purpose and significance.

What do you think? What gives your life meaning?"

# Large Language Models (LLMs)

Text-to-text, translation, code generation



Published as a conference paper at ICLR 2024

SWE-BENCH: CAN LANGUAGE MODELS RESOLVE  
REAL-WORLD GITHUB ISSUES?

Carlos E. Jimenez<sup>\* 1,2</sup> John Yang<sup>\* 1,2</sup> Alexander Wettig<sup>1,2</sup>  
Shunyu Yao<sup>1,2</sup> Kexin Pei<sup>3</sup> Ofir Press<sup>1,2</sup> Karthik Narasimhan<sup>1,2</sup>

<sup>1</sup>Princeton University <sup>2</sup>Princeton Language and Intelligence <sup>3</sup>University of Chicago

# Can LLMs solve real-world GitHub issues?

Published as a conference paper at ICLR 2024

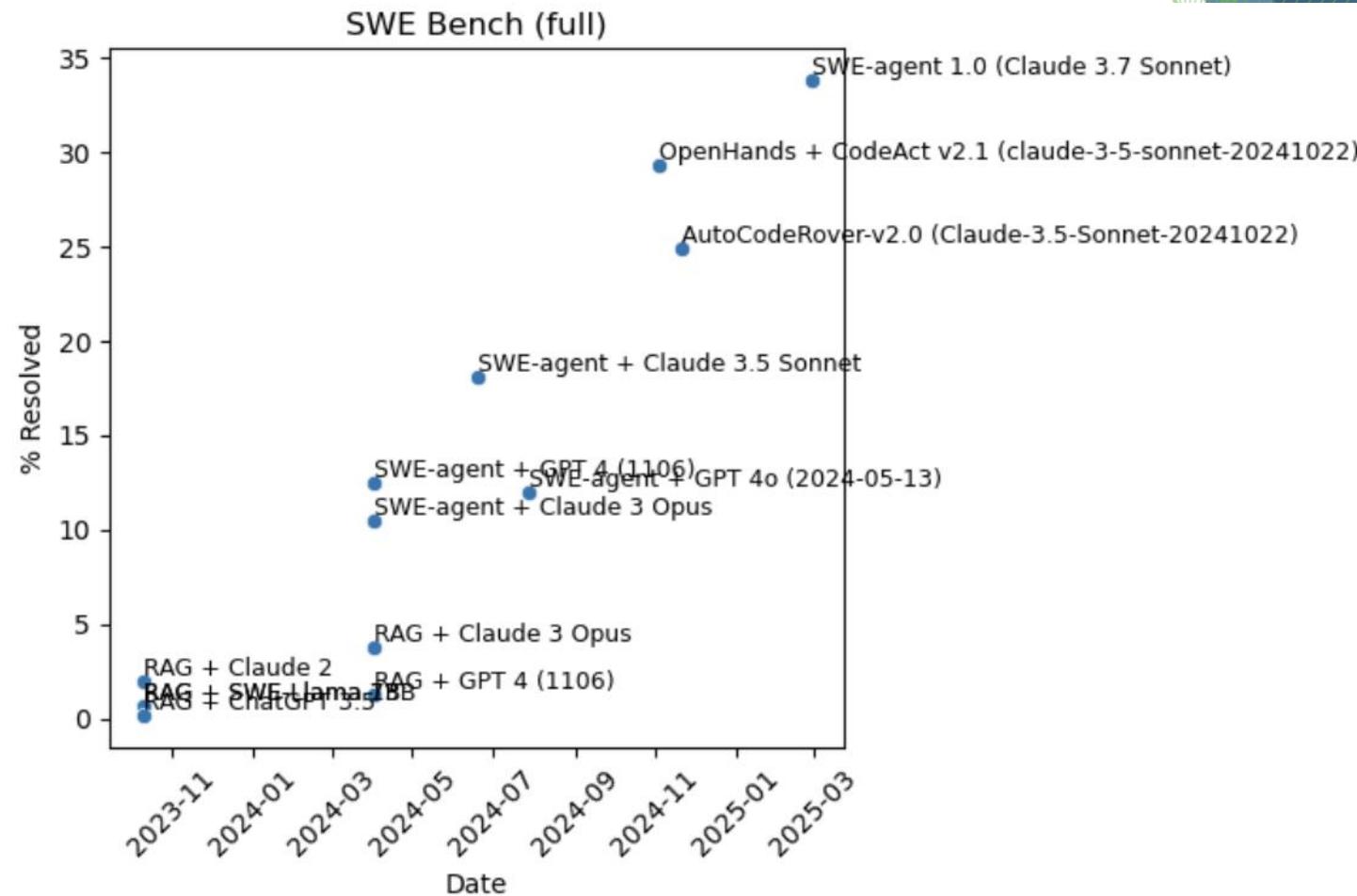
## SWE-BENCH: CAN LANGUAGE MODELS RESOLVE REAL-WORLD GITHUB ISSUES?

Carlos E. Jimenez<sup>\* 1,2</sup> John Yang<sup>\* 1,2</sup> Alexander Wettig<sup>1,2</sup>

Shunyu Yao<sup>1,2</sup> Kexin Pei<sup>3</sup> Ofir Press<sup>1,2</sup> Karthik Narasimhan<sup>1,2</sup>

<sup>1</sup>Princeton University <sup>2</sup>Princeton Language and Intelligence <sup>3</sup>University of Chicago

Model	SWE-bench	
	% Resolved	% Apply
Claude 3 Opus	<b>3.79</b>	46.56
Claude 2	1.97	43.07
ChatGPT-3.5	0.17	26.33
GPT-4-turbo	1.31	26.90
SWE-Llama 7b	0.70	51.74
SWE-Llama 13b	0.70	<b>53.62</b>

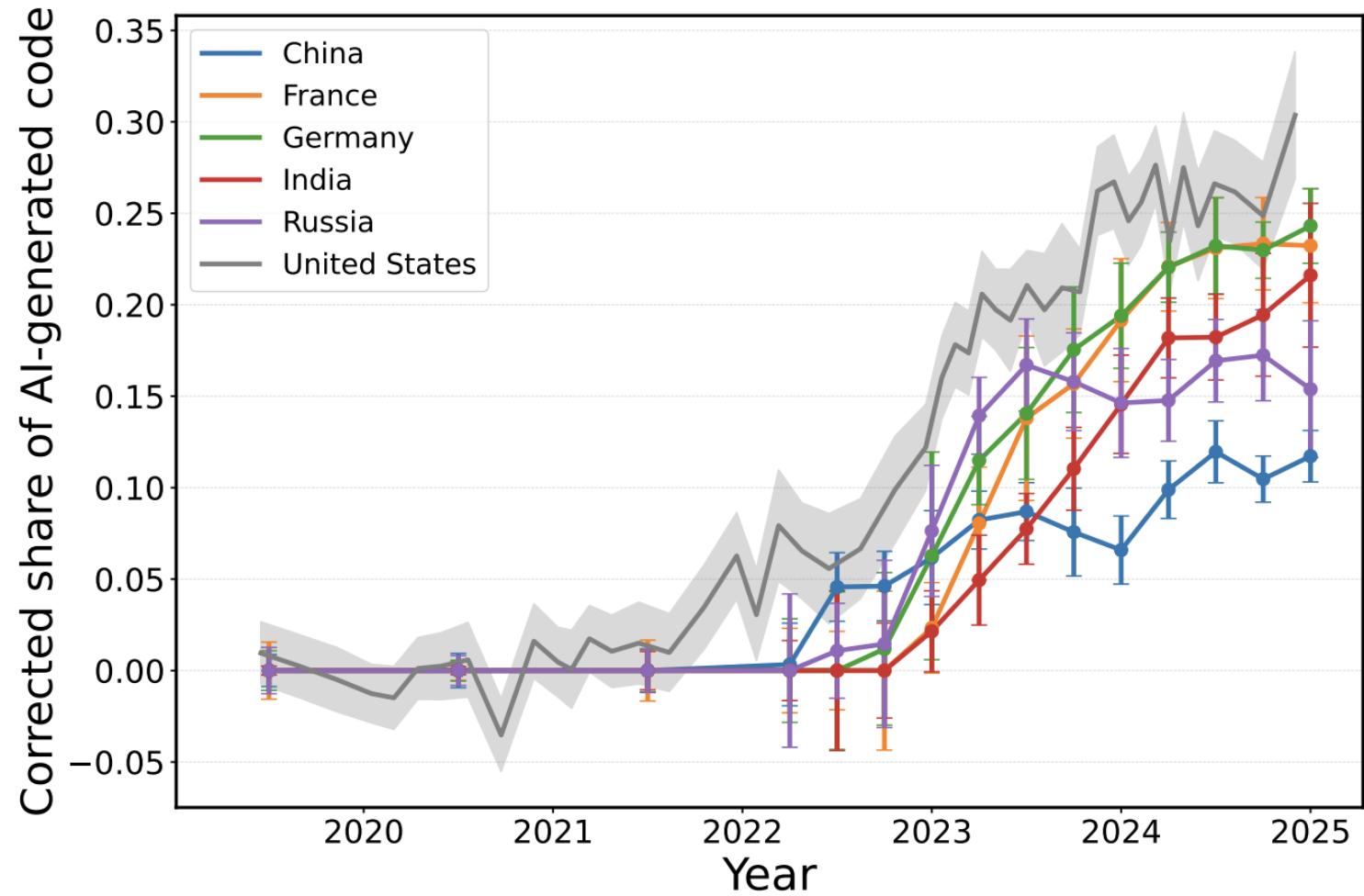


# AI-generated Code

... appears online  
more and more

Indicator: Share of AI-generated code to Python-Contributions on Github.com

(Hint: Presumably not submitted by AI-agents, but by humans using ChatGPT)

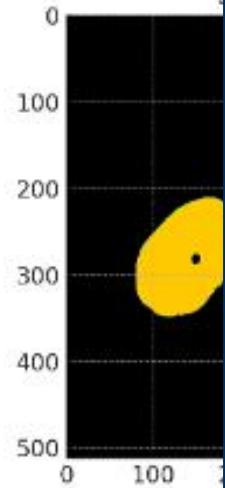
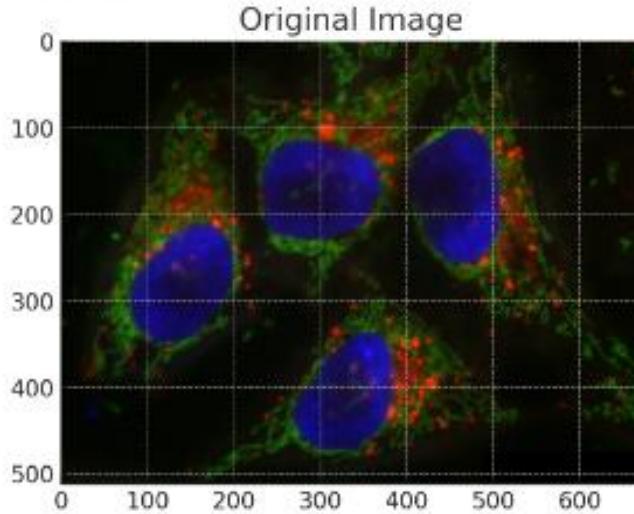


# Image Analysis using Large-Language Models

Analysing images in the browser, e.g. using ChatGPT

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

ChatGPT



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



## Analysis

```
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 channel
blue_channel = image[:, :, 2]

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

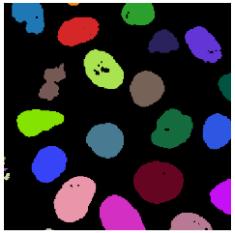
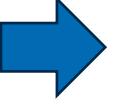
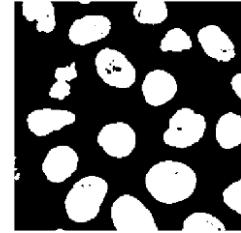
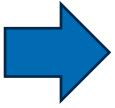
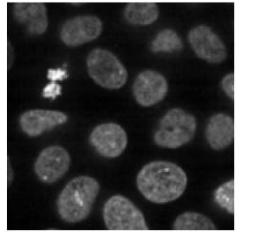
# Remove small objects and perform morphological closing to
# close gaps between nuclei
cleaned = morphology.remove_small_objects(binary, 50)
closed = morphology.closing(cleaned, morphology.disk(3))

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

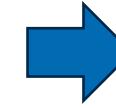
# Plot the original image and the binary image
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
ax[0].imshow(image)
ax[0].set_title('Original Image')
ax[1].imshow(label_image, cmap='nipy_spectral')
ax[1].set_title('Segmented Nuclei')
plt.show()
```

# How good are LLMs for Bio-image Analysis?

Test 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
...	



858.04

workflow\_segmentation\_measurement\_summary

reference	1.0	1.0	1.0	0.9	1.0	0.8	0.9	0.0	0.3	0.4	0.5	0.0	0.6	0.1	0.4	0.4	0.1	0.5	0.0	0.1	0.1
claude-3-5-sonnet-20240620	gpt-4o-2024-05-13	gpt-4-turbo-2024-04-09	claude-3-opus-20240229	gpt-4-1106-preview	gemini-1.5-pro-001	gpt-4o-mini-2024-07-18	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	llama3-8b-instruct-fp16	command-r-plus-104b-q4_0	codellama		

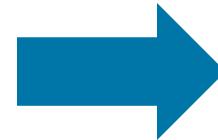
Unit-test pass-rate (n=10):

Large language models

# Benchmarking LLMs for Bio-image Analysis

Use-case: compute the correlation matrix

	label	maximum	mean	median	minimum	sigma	sum	variance	elongation	feret_diameter
0	1	143.0	117.489451	117.041016	93.0	9.489786	27845.0	90.056032	1.228690	8.774964
1	2	113.0	83.052219	82.177734	65.0	9.699808	31809.0	94.086271	1.325096	13.152946
2	3	130.0	108.930403	108.076172	92.0	7.557057	29738.0	57.109109	1.565911	12.884099
3	4	129.0	94.576991	93.134766	70.0	11.433116	53436.0	130.716136	1.227027	14.352700
4	5	149.0	119.454545	119.033203	89.0	12.017958	32850.0	144.431321	1.429829	10.723805
5	6	142.0	124.947368	126.005859	104.0	10.544100	11870.0	111.178052	1.196911	9.273618
6	7	148.0	124.129964	124.013672	99.0	11.807226	34384.0	139.410584	1.137575	13.114877
7	8	152.0	103.128120	99.111328	74.0	18.019024	61980.0	324.685225	1.373404	14.866069
8	9	137.0	83.685407	79.189453	59.0	16.856217	69961.0	284.132050	1.205095	14.456832
9	10	163.0	135.924832	136.962891	99.0	13.564650	101264.0	183.999719	1.169756	14.730920
10	11	175.0	98.692044	92.138672	59.0	25.488803	115371.0	649.679055	1.201427	16.911535
11	12	127.0	76.991909	72.216797	54.0	16.391926	47581.0	268.695234	1.272889	15.066519
12	13	160.0	130.877049	129.990234	109.0	10.845245	63868.0	117.619349	1.143453	13.152946
13	14	123.0	112.602410	112.060547	103.0	4.444618	9346.0	19.754628	2.025953	9.110434
14	15	175.0	123.444924	121.025391	83.0	18.915416	57155.0	357.792956	1.339038	13.892444



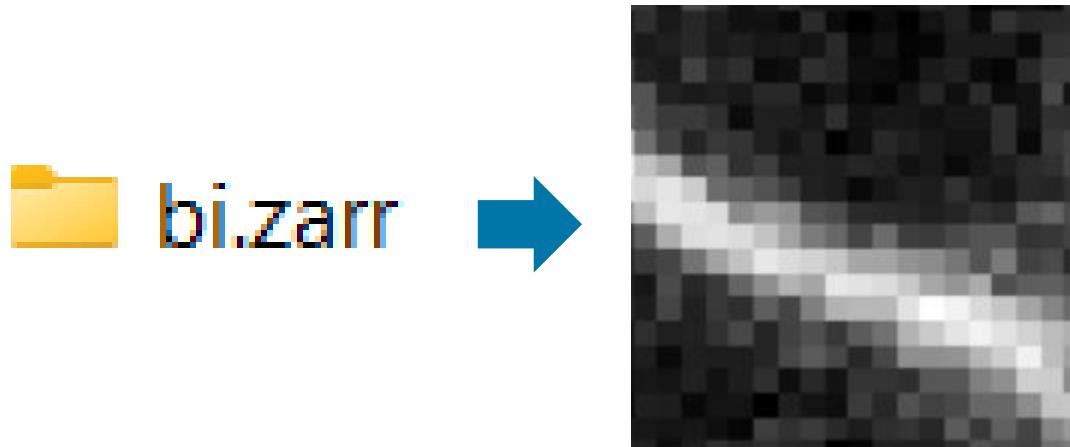
	label	maximum	mean	median	minimum	sigma	sum	variance	elongation	feret_diameter
label	1.000000	-0.605035	-0.651268	-0.637158	-0.581233	-0.136779	-0.170934	-0.134539	-0.014857	0.105859
maximum	-0.605035	1.000000	0.824653	0.787135	0.577706	0.576118	0.522637	0.563160	-0.028076	0.144944
mean	-0.651268	0.824653	1.000000	0.997051	0.918750	0.040456	0.157387	0.052848	0.122359	-0.173872
median	-0.637158	0.787135	0.997051	1.000000	0.928168	-0.013701	0.111119	-0.002710	0.133479	-0.205403
minimum	-0.581233	0.577706	0.918750	0.928168	1.000000	-0.317573	-0.057844	-0.273489	0.217240	-0.311868
sigma	-0.136779	0.576118	0.040456	-0.013701	-0.317573	1.000000	0.633356	0.959845	-0.250655	0.436449
sum	-0.170934	0.522637	0.157387	0.111119	-0.057844	0.633356	1.000000	0.602559	-0.074555	0.775197
variance	-0.134539	0.563160	0.052848	-0.002710	-0.273489	0.959845	0.602559	1.000000	-0.191963	0.370870
elongation	-0.014857	-0.028076	0.122359	0.133479	0.217240	-0.250655	-0.074555	-0.191963	1.000000	0.184445
feret_diameter	0.105859	0.144944	-0.173872	-0.205403	-0.311868	0.436449	0.775197	0.370870	0.184445	1.000000

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

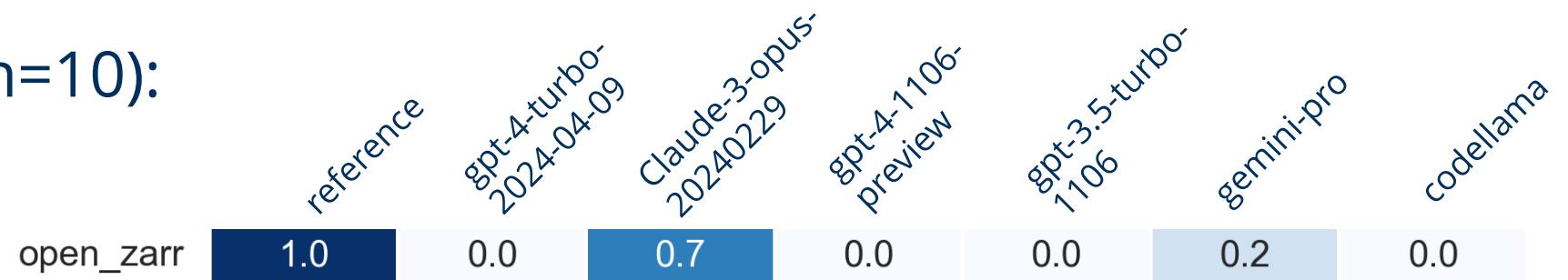


# Benchmarking LLMs for Bio-image Analysis

Use case: Open a zarr file

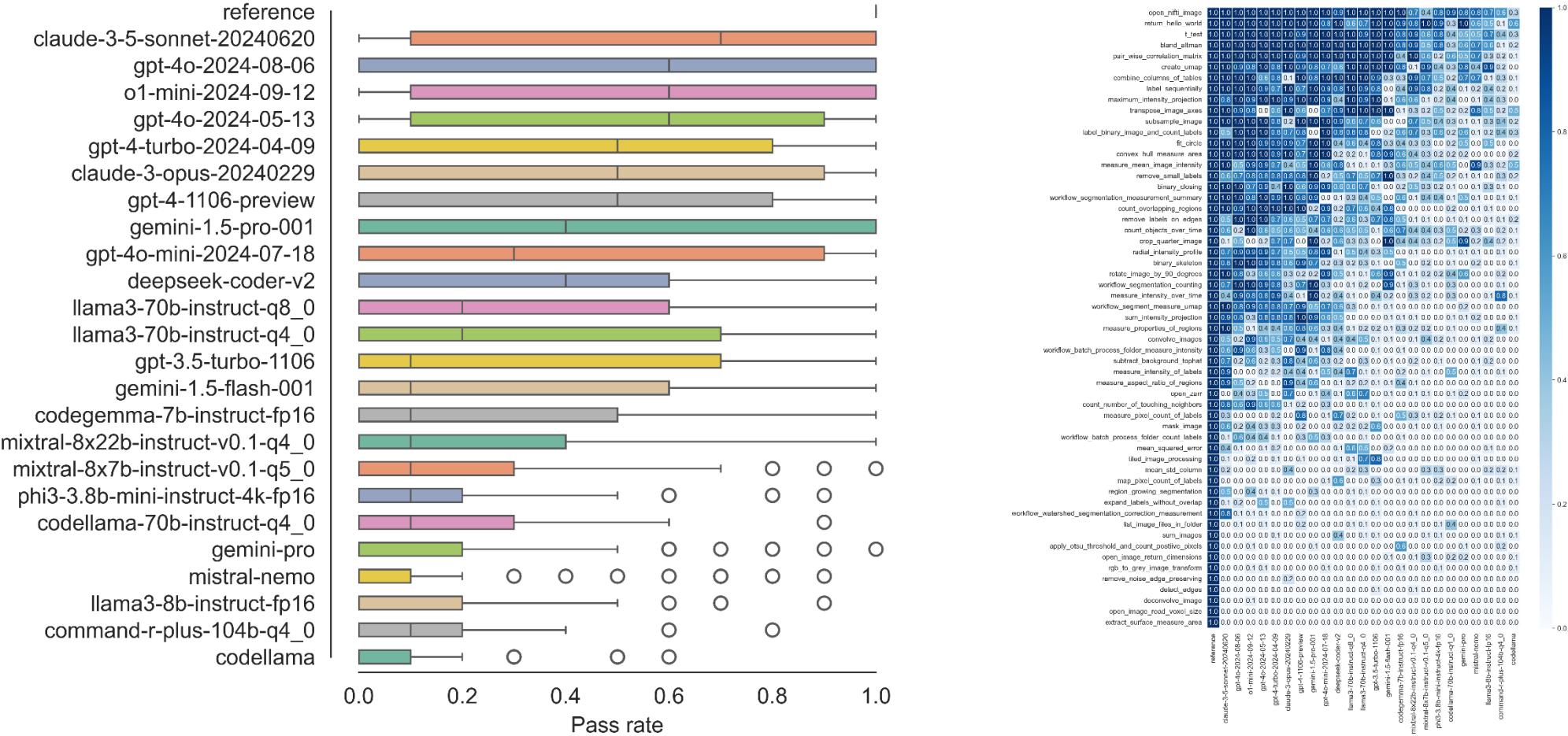


Unit-test pass-rate (n=10):



# Benchmarking LLMs for Bio-image Analysis

Summary: 57 use-cases (yet), 24 LLMs (yet), n=10



# Prompt Engineering

To further improve  
LLMs capabilities

# Prompt Engineering

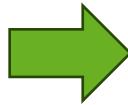
*Modify a prompt until it works.*

Write python code to open the image file 'data/blobs.tif'

and threshold it using Otsu's method.

Apply connected component labeling to the result to

retrieve a label image. Visualize the label image.



Here is the python code to achieve the task:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image
img = cv2.imread('data/blobs.tif', 0)

# Apply Otsu's thresholding
_, thresholded = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Apply connected component labeling
num_labels, labels = cv2.connectedComponents(thresholded)

# Visualize the label image
plt.figure(figsize=(10, 10))
plt.imshow(labels, cmap='nipy_spectral')
plt.colorbar()
plt.show()
```

Make sure you have the necessary libraries (OpenCV, NumPy, and Matplotlib) installed in your Python environment before running the code.

# Prompt Engineering

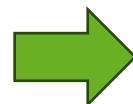
*Modify a prompt until it works.*

Write python code to open the image file 'data/blobs.tif'  
and threshold it using Otsu's method.

Apply connected component labeling to the result to  
retrieve a label image. Visualize the label image.

Do not explain the solution.

Only write Python code and short comments are ok.



```
import cv2
import numpy as np

# Read the image
img = cv2.imread('data/blobs.tif', cv2.IMREAD_GRAYSCALE)

# Apply Otsu's thresholding
_, thresh = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Apply connected component labeling
num_labels, label_image = cv2.connectedComponents(thresh)

# Visualize the label image
cv2.imshow('Label Image', label_image.astype(np.uint8) * 50)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

# Prompt Engineering

*Modify a prompt until it works.*

„Long“ Context

Write python code to open the image file 'data/blobs.tif' and threshold it using Otsu's method.

Apply connected component labeling to the result to retrieve a label image. Visualize the label image.

Do not explain the solution.

Only write Python code and short comments are ok.

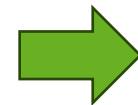
Assume the code is executed in a Jupyter notebook.

Do not use any of these Python libraries:

\* OpenCV

\* PIL

\* Pillow



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

# Load the image
image = io.imread('data/blobs.tif')

# Threshold using Otsu's method
thresh = filters.threshold_otsu(image)
binary_image = image > thresh

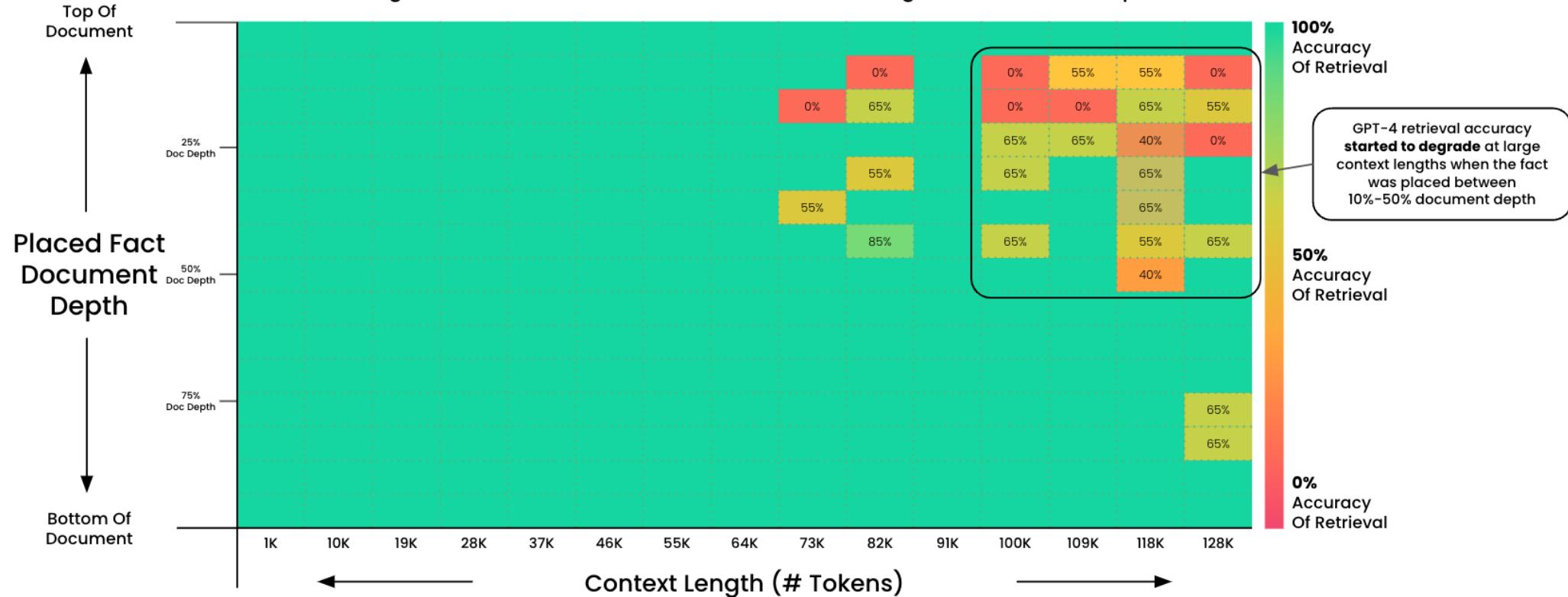
# Connected component labeling
label_image = measure.label(binary_image)

# Visualize the label image
plt.figure(figsize=(8,8))
plt.imshow(label_image, cmap='nipy_spectral')
plt.axis('off')
plt.show()
```

# Prompt engineering

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

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



### Goal: Test GPT-4 Ability To Retrieve Information From Large Context Windows

A fact was placed within a document. GPT-4 (1106-preview) was then asked to retrieve it. The output was evaluated for accuracy.

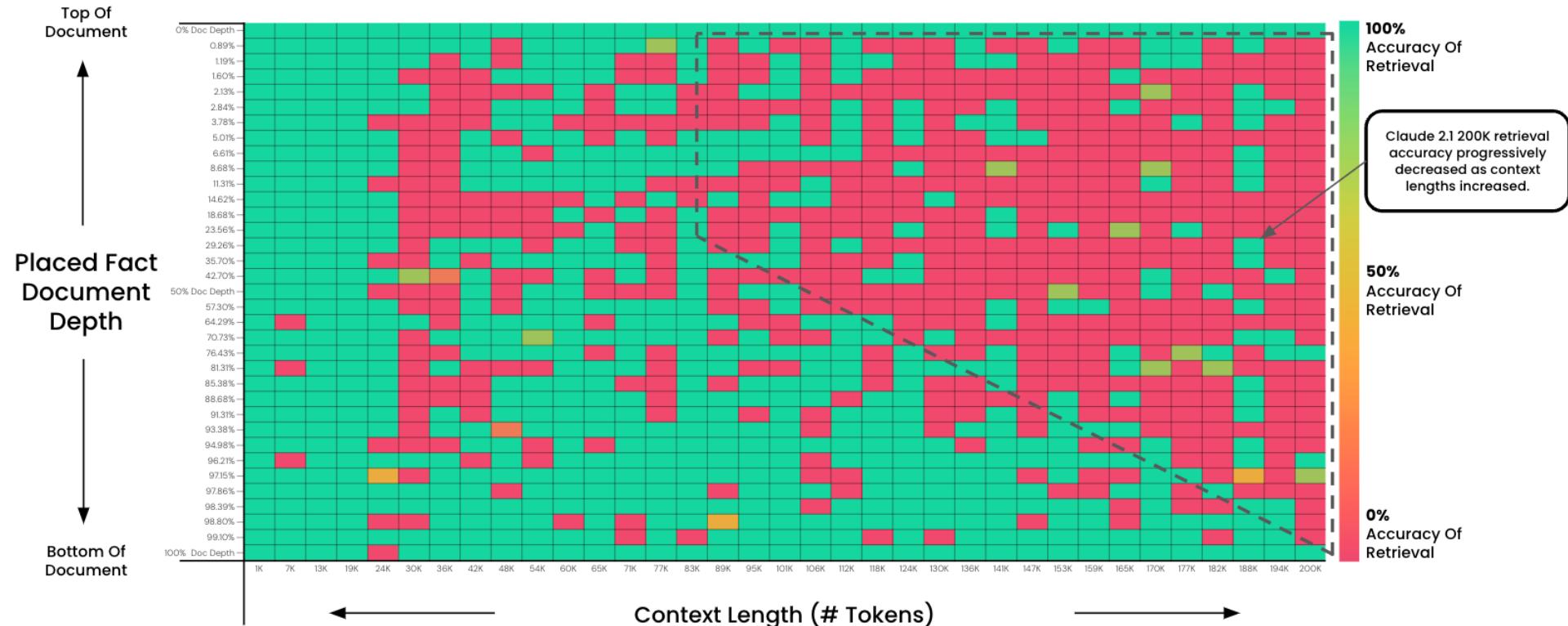
This test was run at 15 different document depths (top > bottom) and 15 different context lengths (1K > 128K tokens).

2x tests were run for larger contexts for a larger sample size.

# Prompt engineering

## Pressure Testing Claude-2.1 200K via "Needle In A HayStack"

Asking Claude 2.1 To Do Fact Retrieval Across Context Lengths & Document Depth



### Goal: Test Claude 2.1 Ability To Retrieve Information From Large Context Windows

A fact was placed within a document. Claude 2.1 (200K) was then asked to retrieve it. The output was evaluated (with GPT-4) for accuracy. This test was run at 35 different document depths (top > bottom) and 35 different context lengths (1K > 200K tokens).

Document Depths followed a sigmoid distribution

# Prompt engineering in practice

```
knowledge_base = """## Code snippets
```

You typically assemble code snippets when working with Python. The following snippets are given:

- \* When asked to segment nuclei, use the cellpose library like this:

```
```python
from cellpose import models

model = models.CellposeModel(gpu=False)
masks, flows, styles = model.eval(image,
                                    batch_size=32,
                                    flow_threshold=0.4,
                                    cellprob_threshold=0.0,
                                    normalize={"tile_norm_blocksize": 0})
label_image = masks.astype(np.uint32)
```

```

- \* When asked to segment anything else but nuclei, use otsu-thresholding:

```
```python
from skimage.measure import label
from skimage.filters import threshold_otsu

```

Such knowledge bases can be *hundreds* of examples, *thousands* of tokens

# Scientific code writing: bia-bob



- First steps: Keep it simple

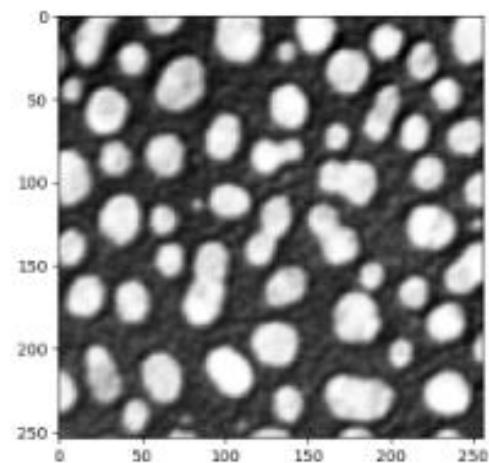
```
%bob Load the blobs.tif image file and show it.
```

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

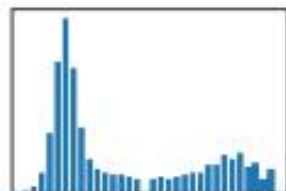
```
from skimage.io import imread
import stackview

# 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



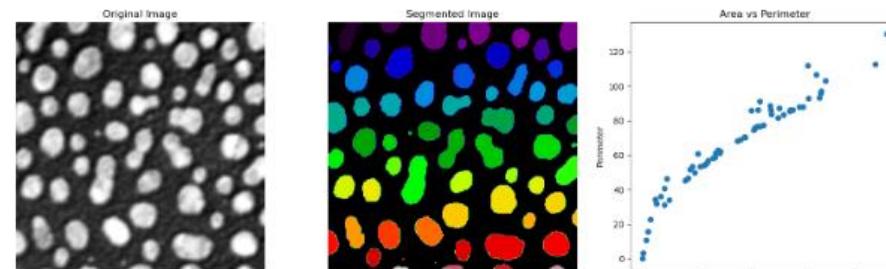
# Scientific code writing: bia-bob



- After learning *how to prompt* for code: try more complex workflows

```
[1]: import bia_bob  
  
[2]: %%bob  
* Load blobs.tif from disk  
* Segment the bright objects on the dark background  
* Measure area and perimeter of the objects  
* Show the original image, the segmented image and an area versus perimeter plot side-by-side using subfigures.
```

This notebook may contain text, code and images generated by artificial intelligence. Used model: claude-3-5-sonnet-20240620, vision model: claude-3-5-sonnet-20240620, endpoint: None, bia-bob version: 0.21.0. Do not enter sensitive or private information and verify generated contents according to good scientific practice. Read more: <https://github.com/haesleinhuepf/bia-bob#disclaimer>



```
import numpy as np  
from skimage import io, filters, measure, segmentation  
import matplotlib.pyplot as plt  
import pandas as pd  
import pyclesperanto_prototype as cle  
import stackview  
  
# Load the image  
image = io.imread('blobs.tif')  
  
# Segment the bright objects  
thresh = filters.threshold_otsu(image)  
binary = image > thresh  
  
# Label the objects  
labels = measure.label(binary)  
  
# Measure properties  
properties = ['label', 'area', 'perimeter']  
df = pd.DataFrame(measure.regionprops_table(labels, properties=properties))  
  
# Create the figure with subplots  
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 5))  
  
# Original image  
ax1.imshow(image, cmap='gray')  
ax1.set_title('Original Image')  
ax1.axis('off')  
  
# Segmented image  
ax2.imshow(labels, cmap='nipy_spectral')  
ax2.set_title('Segmented Image')  
ax2.axis('off')  
  
# Scatter plot  
ax3.scatter(df['area'], df['perimeter'])  
ax3.set_xlabel('Area')  
ax3.set_ylabel('Perimeter')  
ax3.set_title('Area vs Perimeter')  
  
plt.tight_layout()  
plt.show()  
  
# Display the dataframe  
print(df.head())
```

# Notebook generation

```
[2]: %%bob generate a jupyter notebook that
* loads data/blobs.tif
* segments the bright blobs using thresholding and connected component
* measures the area of the objects
* prints out the average area
```

A notebook has been saved as `blob_segmentation_and_area_analysis.ipynb`.

The figure consists of three side-by-side screenshots of Jupyter notebooks. The leftmost screenshot shows the generated notebook's landing page with a title 'Blob Analysis Notebook' and a list of steps. The middle screenshot shows step 3: 'Label connected components', which includes code to load labels from a binary image and a preview of the labeled blobs. The rightmost screenshot shows step 4: 'Measure object areas', which includes code to calculate the area of each blob and a table of results:

label	area
0	433.0
1	185.0
2	658.0
3	434.0
4	477.0

# Optimizing documentation

E.g. to make code more readable

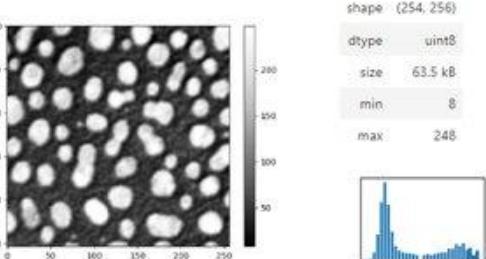


```
[1]: from skimage.io import imread
      import stackview

[2]: image = imread("../blobs.tif")

[3]: stackview.insight(image)

[4]:
```



```
from bia_bob import bob

%%bob modify the notebook `mod_notebook.ipynb` by
adding explanatory markdown cells between the code cells.
Also add a markdown cell at the very beginning that
introduces the entire notebook shortly.

The modified notebook has been saved as mod_notebook_1.ipynb.
```

**Introduction**  
In this notebook, we will demonstrate how to load a biimage, visualize it using stackview, and explore the image interactively. We will be using an image of blobs for this example. The steps include loading the image and visualizing it using different stackview functionalities.

**Step 1: Import necessary libraries**  
We will start by importing the necessary libraries. In this case, `skimage.io` for loading the image and `stackview` for visualization.

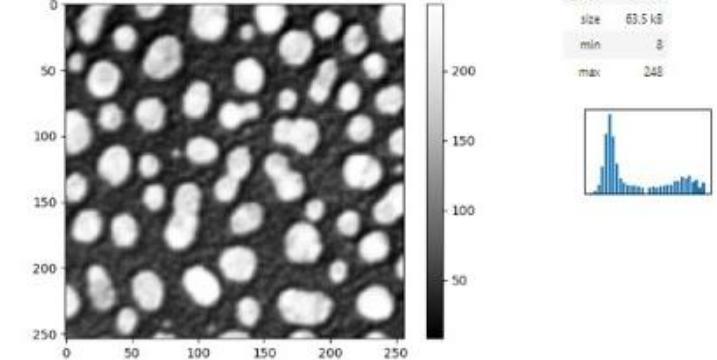
```
from skimage.io import imread
import stackview
```

**Step 2: Load the image**  
Next, we will load the image file into a variable. The image used in this notebook is `blobs.tif`.

```
image = imread("../blobs.tif")
```

**Step 3: Visualize the image using stackview insight**  
We will use `stackview.insight` to visualize the loaded image. This function allows for inspecting the image.

```
stackview.insight(image)
```



# Translating notebooks

%bob Please  
translate  
notebook  
`<xyz.ipynb>` to  
`<language>`.

The figure displays four side-by-side Jupyter notebook interfaces, each showing a different language version of a Python script for image processing. The notebooks are titled with the names of the target languages: English, German, Spanish, and French. Each notebook contains code snippets, explanatory text, and visual outputs (e.g., brain MRI slices) identical across all four versions.

- English:** Titled "Multidimensional image stacks".
  - Text:** Multidimensional image data can be handled in a similar multi-channel image data.
  - Code:**

```
from skimage.io import imread
from stackview import imshow
import matplotlib.pyplot as plt
image_stack = imread('../data/Haase_MRT_tfl3d1.tif')
```
  - Image:** Three orthogonal slices of a brain MRI dataset (Sagittal, Coronal, Transverse).
  - Text:** If an image dataset has a temporal dimension, we call it a video. Processing videos works similar to multi-channel images and image stacks. Let's open a microscopy dataset showing yeast.
- German:** Titled "Multidimensionale Bildstapel".
  - Text:** Multidimensionale Bilddaten können ähnlich wie mehrkanalige Bilddaten behandelt werden.
  - Code:**

```
from skimage.io import imread
from stackview import imshow
import matplotlib.pyplot as plt
image_stack = imread('../data/Haase_MRT_tfl3d1.tif')
```
  - Image:** Three orthogonal slices of a brain MRI dataset (Sagittal, Coronal, Transverse).
  - Text:** Es gibt auch Bilder mit drei räumlichen Dimensionen: X, Y und Z. Typische Beispiele finden sich in der Mikroskopie und in medizinischen Bildgebungen. Schauen wir uns ein Magnetresonanztomographie (MRT) Datensatz an:
- Spanish:** Titled "Piles de imágenes multidimensionales".
  - Text:** Los datos de imágenes multidimensionales se pueden manejar de manera similar a los datos de imágenes multicanal.
  - Code:**

```
from skimage.io import imread
from stackview import imshow
import matplotlib.pyplot as plt
image_stack = imread('../data/Haase_MRT_tfl3d1.tif')
```
  - Image:** Three orthogonal slices of a brain MRI dataset (Sagittal, Coronal, Transverse).
  - Text:** También hay imágenes con tres dimensiones especiales: X, Y y Z. Puedes encontrar ejemplos típicos en microscopio y en imágenes médicas. Echemos un vistazo a un conjunto de datos de imágenes por Resonancia Magnética (IRM):
- French:** Titled "Piles d'images multidimensionnelles".
  - Text:** Les données d'images multidimensionnelles peuvent être traitées de manière similaire aux données d'images multi-canaux.
  - Code:**

```
from skimage.io import imread
From stackview import imshow
import matplotlib.pyplot as plt
image_stack = imread('../data/Haase_MRT_tfl3d1.tif')
```
  - Image:** Three orthogonal slices of a brain MRI dataset (Sagittal, Coronal, Transverse).
  - Text:** Il existe aussi des images avec trois dimensions spatiales : X, Y et Z. Vous trouvez des exemples typiques en microscope et en imagerie médicale. Regardons un ensemble de données d'imagerie par résonance magnétique (IRM) :

# Exercises

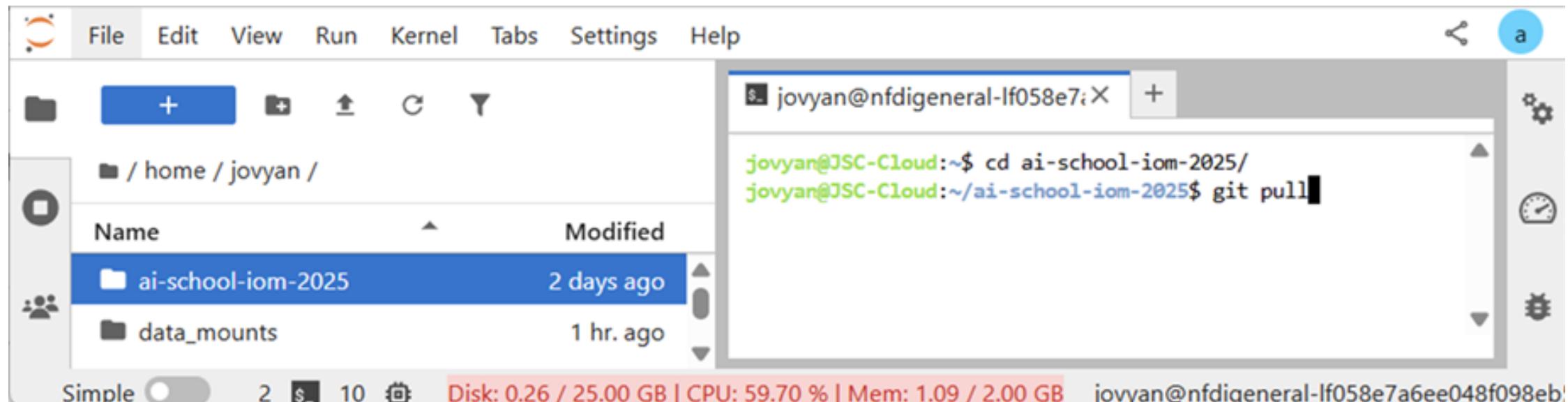
# Login to jupyter4nfdi

Navigate to <https://hub.nfdi-jupyter.de/>, log in and start a Jupyter Lab.

If you still have the ai-school-iom-2025 folder, enter this in a terminal:

```
cd ai-school-iom-2025
```

```
git pull
```



# Login to jupyter4nfdi

Navigate to <https://hub.nfdi-jupyter.de/>, log in and start a Jupyter Lab.

If you DO NOT have the ai-school-iom-2025 folder, enter this in a terminal:

```
git clone https://github.com/ScaDS/ai-school-iom-2025/
```

## In case of problems

```
mkdir test2
```

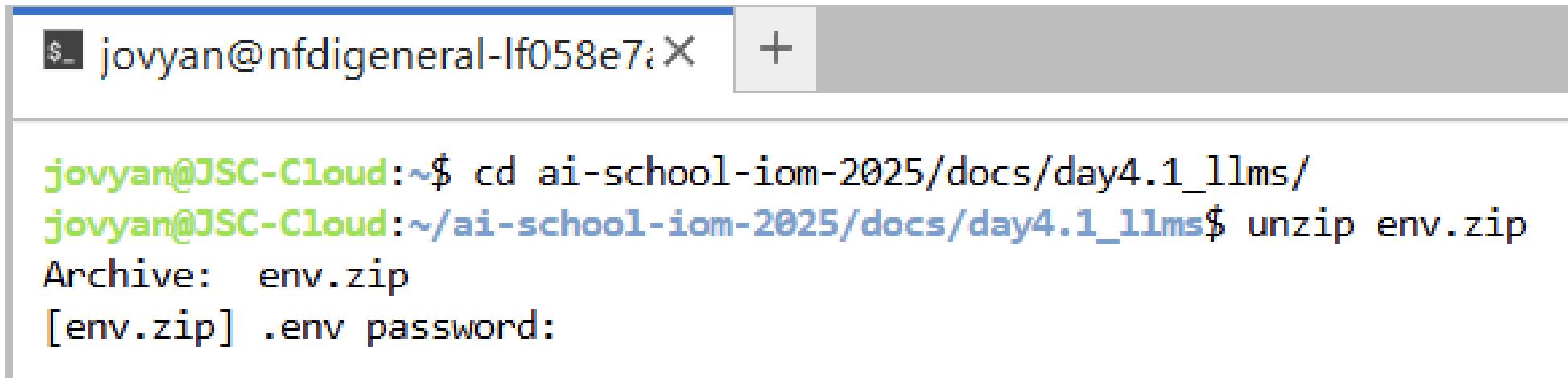
```
cd test2
```

```
git clone https://github.com/ScaDS/ai-school-iom-2025/
```

# Unzip the env.zip file

Navigate to the folder `ai-school-iom-2025/docs/day4.1_llms` and unzip the `env.zip` file:

```
cd ai-school-iom-2025/docs/day4.1_llms/  
unzip env.zip
```

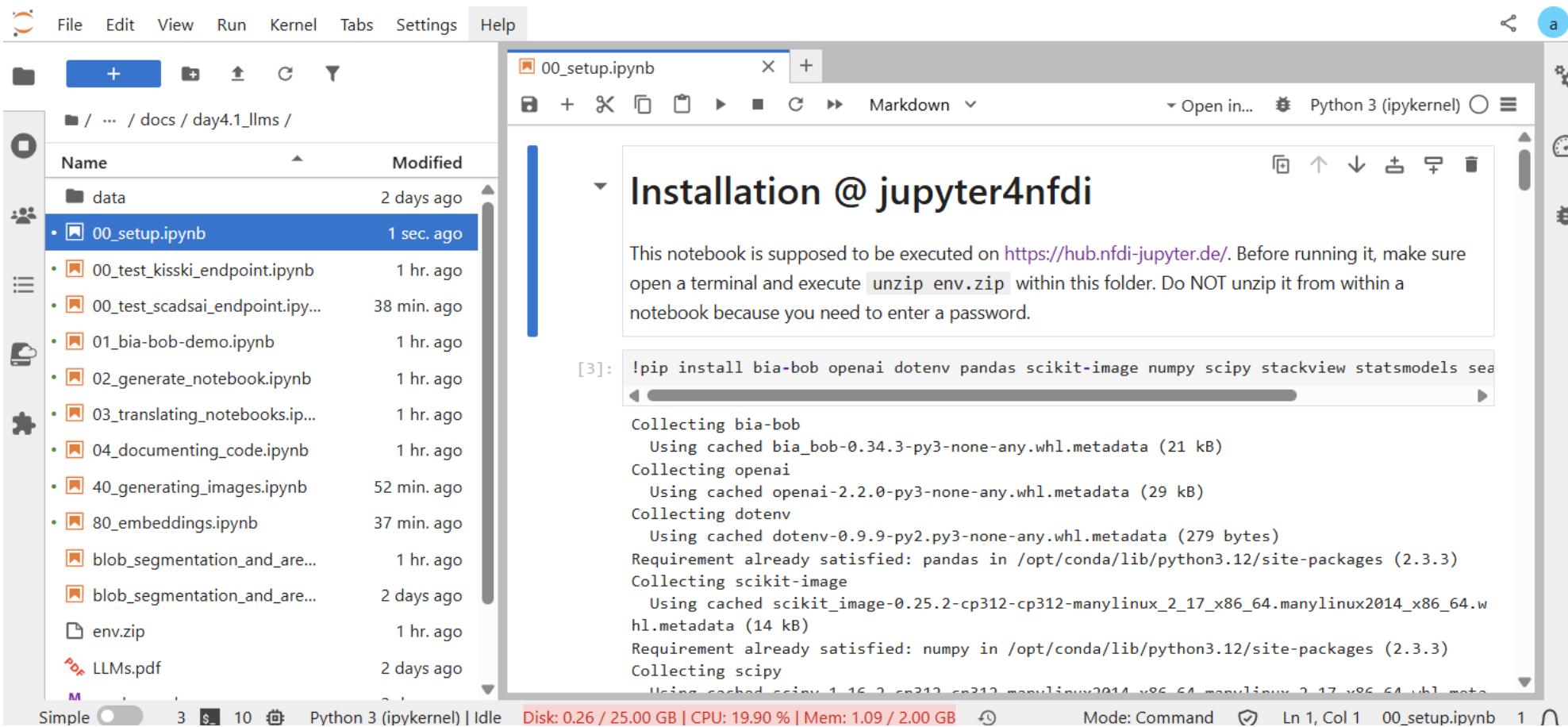


A screenshot of a terminal window. The title bar shows the user's name and session ID: `jovyan@nfdigeneral-1f058e7:~$`. Below the title bar, there is a plus sign button for opening new tabs. The main area of the terminal displays the following command and its execution:

```
jovyan@JSC-Cloud:~$ cd ai-school-iom-2025/docs/day4.1_llms/  
jovyan@JSC-Cloud:~/ai-school-iom-2025/docs/day4.1_llms$ unzip env.zip  
Archive: env.zip  
[env.zip] .env password:
```

# Install requirements

First, run the notebook `docs/day4.1_llms/00_setup.ipynb`



The screenshot shows a Jupyter Notebook interface with a sidebar on the left containing a file tree. The file tree shows a folder structure under `/ ... / docs / day4.1_llms /`, with several IPython notebooks listed. One notebook, `00_setup.ipynb`, is selected and highlighted in blue. The main pane displays the content of this notebook, which includes a section titled **Installation @ jupyter4nfdi**. Below this, there is a note about executing the notebook on <https://hub.nfdi-jupyter.de/> and running `!pip install bia-bob openai dotenv pandas scikit-image numpy stackview statsmodels sea`. The notebook then shows the output of this command, which lists various packages being installed from cached sources. At the bottom of the notebook, there is a status bar showing system information like disk usage (Disk: 0.26 / 25.00 GB), CPU usage (CPU: 19.90 %), memory usage (Mem: 1.09 / 2.00 GB), and the current mode (Mode: Command). The status bar also indicates the current file is `00_setup.ipynb`.

# The the connection(s) to the LLM server(s)

ScaDS.AI LLM endpoint

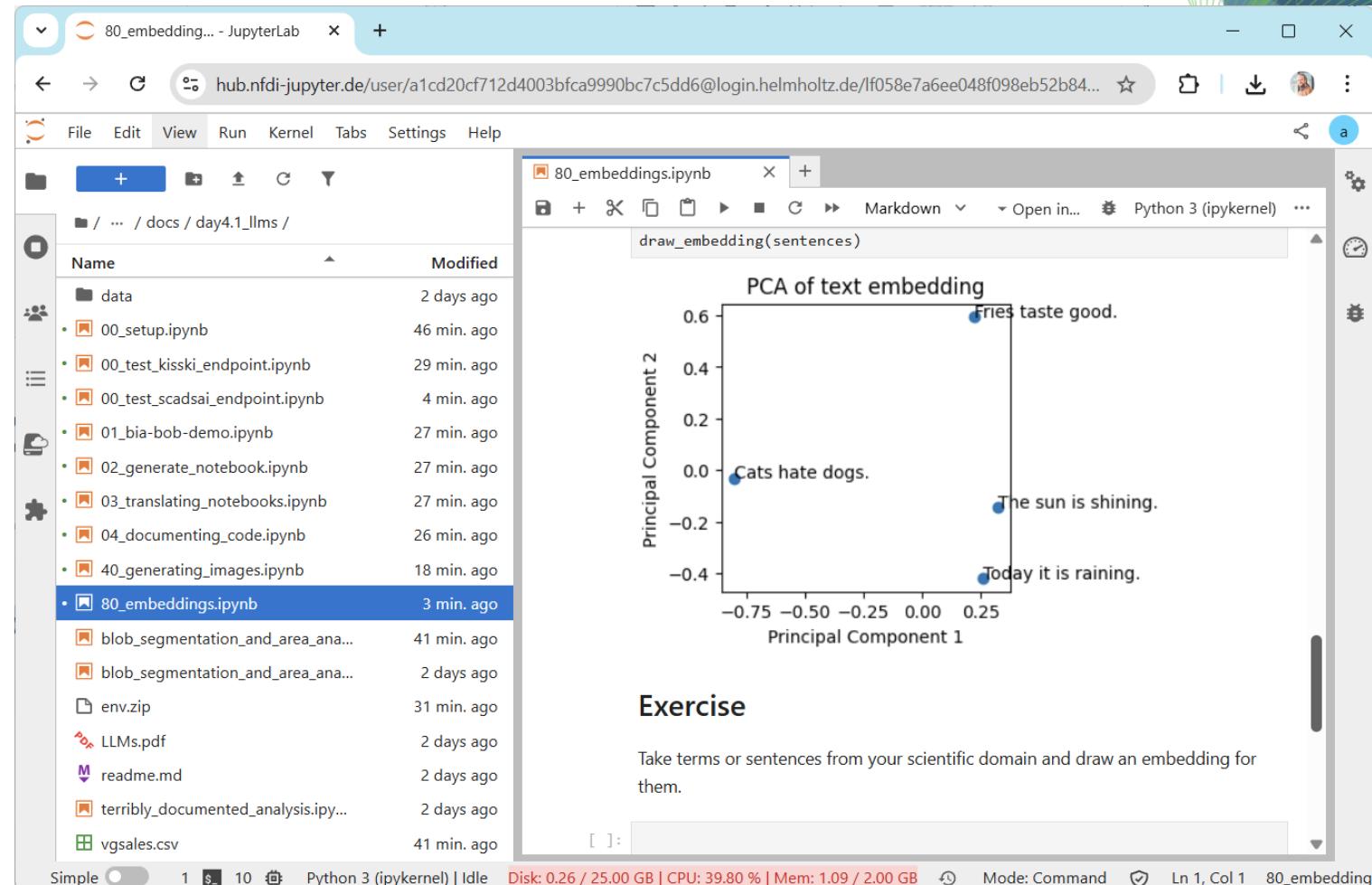
In this notebook we will use the ScaDS.AI LLM infrastructure at the Center for

```
[10]: prompt_scadsai_llm("Hi!")
```

```
[10]: 'How are you doing today? Is there something I can help you with, or would you like to chat?'
```

# Embeddings

Draw embeddings of words / sentences from your own scientific domain



## Exercise

Take terms or sentences from your scientific domain and draw an embedding for them.

# Bia-Bob: code generation

The image shows a dual-monitor setup. On the left monitor, a web browser displays the BiA-Bob documentation at [scads.github.io/EMBOBiolImage2025/02\\_bia\\_bob/bia-bob-demo.html](https://scads.github.io/EMBOBiolImage2025/02_bia_bob/bia-bob-demo.html). The page contains a sidebar with links for Setup, Python Jupyter Notebooks, and AI-assisted Code Generation, including sections on BiA-Bob, Al-assisted Notebook generation, Translating notebooks, Statistical testing, and Under the hood: Configure bia-bob's behaviour through system messages. The main content area is titled "BiA-Bob" and describes the library as a Python interface to large language models. It includes a code snippet for initializing the library:

```
# Load secret API key. You must unpack the contents of
# into the same folder before going ahead.
from dotenv import load_dotenv
load_dotenv()

import os
from bia_bob import bob
bob.initialize(endpoint='https://llm.scads.ai/v1', mode=
```

On the right monitor, a JupyterLab session is open at [localhost:8888/lab/tree/docs/02\\_bia\\_bob/bia-bob-demo.ipynb](http://localhost:8888/lab/tree/docs/02_bia_bob/bia-bob-demo.ipynb). The file tree shows a directory structure with files like data, api\_key.zip, and several .ipynb notebooks. The notebook "bia-bob-demo.ipynb" is currently selected. In the code editor, the following command is written:

```
[3]: %%bob
load data/blobs.tif and show it
```

# Bia-Bob: notebook generation

The screenshot shows a Jupyter Notebook environment with three main windows. The left window, titled 'generate\_notebook.ipynb', contains a single code cell (cell 2) with the following Python code:

```
%bob generate a jupyter notebook that
* loads data/blobs.tif
* segments the bright blobs using thresholding and connected components
* measures the area of the objects
* prints out the average area
```

Below this cell, a message indicates that a notebook has been saved:

A notebook has been saved as `blob_segmentation_and_area_analysis.ipynb`.

The middle window, titled 'blob\_segmentation\_and\_area\_analysis.ipynb', displays a Markdown document with the title 'Blob Analysis' and a section 'Steps' containing a numbered list from 1 to 7. It also includes a code cell showing the import of libraries and the execution of the first few steps.

The right window, also titled 'blob\_segmentation\_and\_area\_analysis.ipynb', shows the continuation of the notebook. It includes sections for '3. Label connected components' and '4. Measure object areas'. The '4. Measure object areas' section contains a code cell that imports `regionprops\_table` and defines properties to measure ('label', 'area'). It then computes measurements and displays the first few rows of the resulting DataFrame. The data is as follows:

label	area
0	433.0
1	185.0
2	658.0
3	434.0
4	477.0

The final section shown is '5. Compute average area', which calculates the mean area of all detected blobs.