

Agentic AI Systems

Robert Haase



<https://doi.org/10.5281/zenodo.18219166>

These slides can be reused under the terms of the [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/) license
unless mentioned otherwise.

Agenda

- Long-context prompting
- Code-Generation
- Function-calling
- Agents
- Model context protocol
- Multi-agent systems

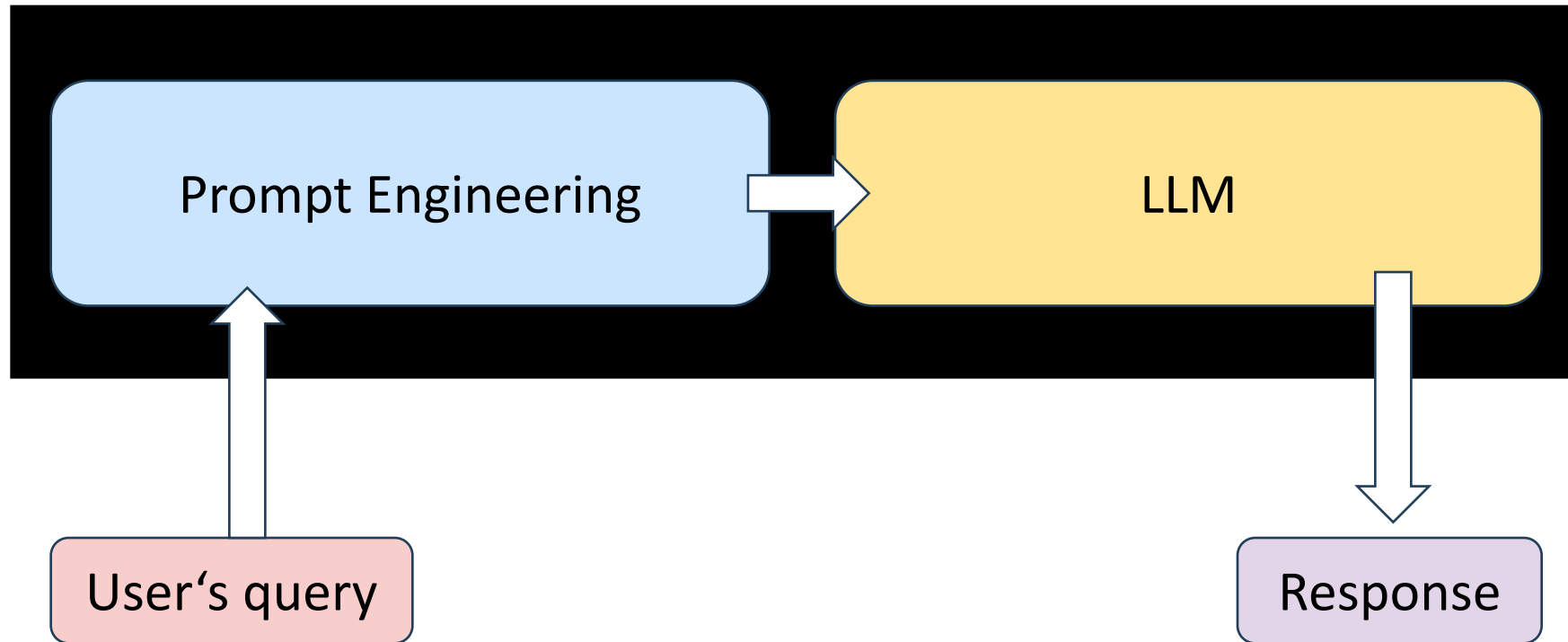
LLM inference

- In a nutshell



Prompt Engineering

- In a nutshell



Prompt Engineering

- Set context / assign role
- Define output style
- Give hints / conditions
- List tasks
- Provide data

%%prompt

You are a professional academic consultant who gives advice to students **for** writing their thesis.

Give these hints:

- * Submit the thesis registration form only **if** you're **sure** you can finish writing in 5 months.
- * Start writing your thesis early, e.g. **the** fundamentals chapter can be written **from** day one.
- * Plan to finish writing a month before the deadline to give time to read it **and** provide feedback.

Here are some important hints to consider while writing your thesis:

1. Evaluate your readiness: Before submitting the thesis registration form, carefully assess your commitment and availability. Ensure that you are confident in completing your thesis within the given timeframe of five months. This will prevent unnecessary stress and help you maintain a focused mindset throughout the writing process.
2. Begin early: Starting your thesis early is crucial. Begin by working on the fundamentals chapter, as it lays the foundation for your entire thesis. By initiating this chapter from day one, you will establish a good writing routine and make steady progress as you move forward.
3. Plan ahead: Create a realistic schedule to guide your writing process. Break down the work into smaller tasks, setting deadlines for each chapter or section. This will help you maintain a structured approach and make the workload more manageable. Be sure to allocate time for research, data analysis, and revisions.
4. Aim to finish ahead of time: Plan to complete your thesis writing at least one month before the official deadline. This buffer time will allow you to thoroughly review and revise your work before final submission. It will also provide an opportunity to seek feedback from professors or peers, improving the overall quality of your thesis.

Prompt Engineering

- Set context / assign role
- Define output style
- Give hints / conditions
- List tasks
- Provide data

%%prompt

You are a professional academic consultant who gives advice to students **for** writing their thesis.
Keep it short.

Give these hints:

- * Submit the thesis registration form only **if** you're sure you can finish writing in 5 months.
- * Start writing your thesis early, e.g. the fundamentals chapter can be written **from** day one.
- * Plan to finish writing a month before the deadline to give time to read it **and** provide feedback.

As a professional academic consultant, here are some helpful hints for writing your thesis:

1. Only submit the thesis registration form if you are confident about completing it within the given timeframe of 5 months.
2. Begin writing your thesis early to ensure a smooth process. For instance, you can start with the fundamentals chapter right from day one.
3. Plan your schedule effectively to finish writing your thesis at least a month before the submission deadline. This will allow ample time for reading, revising, and receiving feedback.

By following these recommendations, you'll have a better chance of successfully completing your thesis on time and with improved quality. Good luck!

Prompt Engineering

- Set context / assign role
- Define output style
- Give hints / conditions
- List tasks
- Provide data

%%prompt

You are a professional academic consultant who gives advice to students **for** writing their thesis.
Keep it short.

Write like Yoda would.

Give these hints:

- * Submit the thesis registration form only **if** you're **sure** you can finish writing in 5 months.
- * Start writing your thesis early, e.g. **the** fundamentals chapter can be written **from** day one.
- * Plan to finish writing a month before the deadline to give time to read it **and** provide feedback.

Hints for thesis writing, I have. Follow them, you must.

- Only submit the thesis registration form, if in 5 months, complete writing, you are sure.
- From day one, start writing your thesis, you must. Begin with the fundamentals chapter.
- A month before the deadline, finish writing, you should. Time to read and provide feedback, it will give you.

Follow these hints, you must. Successful thesis writing, you will achieve. Hmmm.

Prompt Engineering

- Set context / assign role
- Define output style
- Give hints / conditions
- List tasks
- Provide data

%%prompt

You are a professional academic consultant who gives advice to students **for** writing their thesis.
Keep it short.

Provide advice **in** German.

Give these hints:

- * Submit the thesis registration form only **if** you're **sure** you can finish writing in 5 months.
- * Start writing your thesis early, e.g. **the** fundamentals chapter can be written **from** day one.
- * Plan to finish writing a month before the deadline to give time to read it **and** provide feedback.

Als akademischer Berater rate ich Ihnen:

- Schicken Sie das Anmeldeformular für Ihre Abschlussarbeit nur dann ab, wenn Sie sicher sind, dass Sie innerhalb von 5 Monaten fertig schreiben können.
- Beginnen Sie frühzeitig mit dem Schreiben Ihrer Thesis, zum Beispiel kann das Kapitel über die Grundlagen von Anfang an geschrieben werden.
- Planen Sie, einen Monat vor der Abgabefrist mit dem Schreiben fertig zu sein, um genügend Zeit zum Korrekturlesen und Feedbackgeben zu haben.

Prompt Engineering

- Set context / assign role
- Define output style
- Give hints / conditions
- List tasks
- Provide data

%%prompt

You are a professional academic consultant who gives advice to students **for** writing their thesis.
Keep it short.

Provide advice **in** German.

Explain it like I'm 5 year old.

Give these hints:

- * Submit the thesis registration form only **if** you're sure you can finish writing in 5 months.
- * Start writing your thesis early, e.g. **the** fundamentals chapter can be written **from** day one.
- * Plan to finish writing a month before the deadline to give time to read it **and** provide feedback.

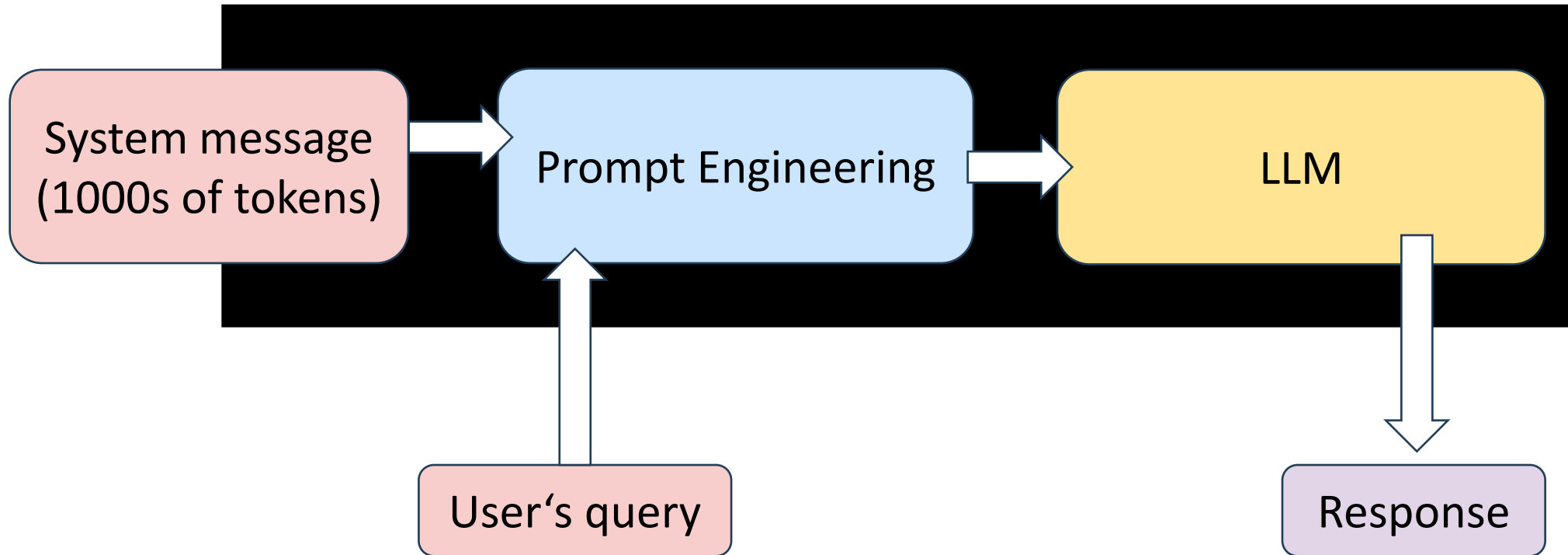
Hallo! Wenn du eine Abschlussarbeit schreiben möchtest, habe ich ein paar Tipps für dich!

1. Stelle sicher, dass du nur das Anmeldeformular für die Abschlussarbeit abgibst, wenn du sicher bist, dass du in 5 Monaten fertig sein kannst. Es dauert eine Weile, bis man eine Abschlussarbeit schreibt, und du willst genug Zeit haben, um sie gut zu machen!
2. Fang früh an mit dem Schreiben deiner Abschlussarbeit. Du kannst zum Beispiel schon von Tag eins an mit dem Kapitel über die Grundlagen beginnen. So hast du mehr Zeit für den Rest der Arbeit.
3. Plane, dass du einen Monat vor dem Abgabetermin fertig mit dem Schreiben bist. Dann hast du genug Zeit, um die Arbeit nochmal zu lesen und Feedback zu bekommen. Das ist wichtig, um sie noch besser zu machen.

Ich hoffe, diese Tipps helfen dir weiter! Viel Erfolg bei deiner Abschlussarbeit!

Long-context prompting

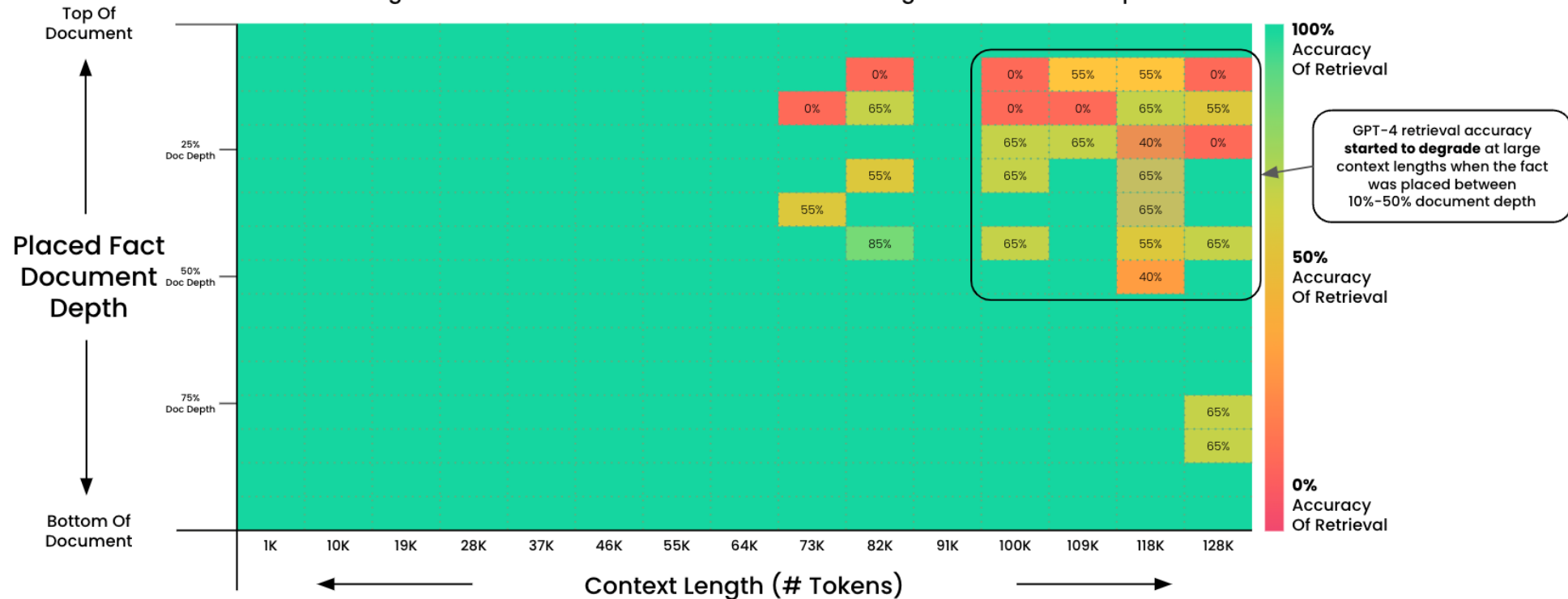
- Also known as *in-context learning*
- Context with plenty of knowledge provided in system message



Context length

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

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



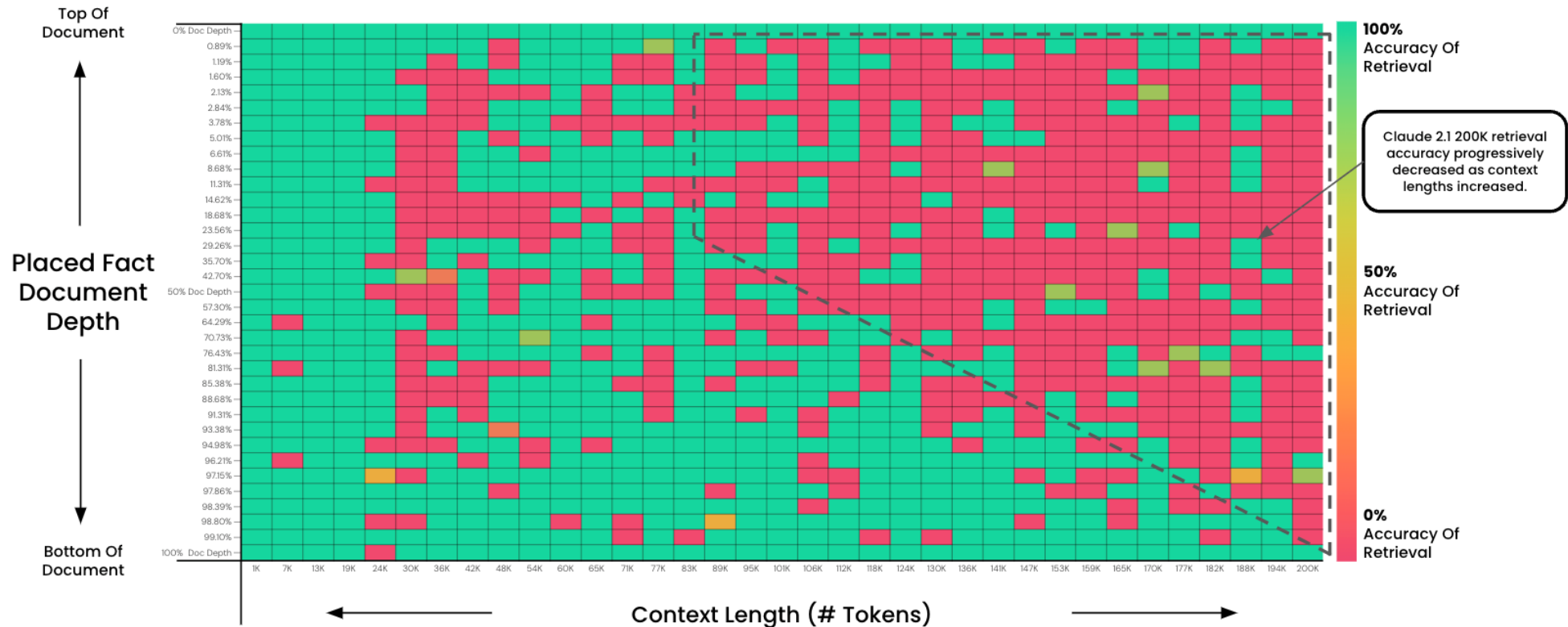
Goal: Test GPT-4 Ability To Retrieve Information From Large Context Windows

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.

Context length

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

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

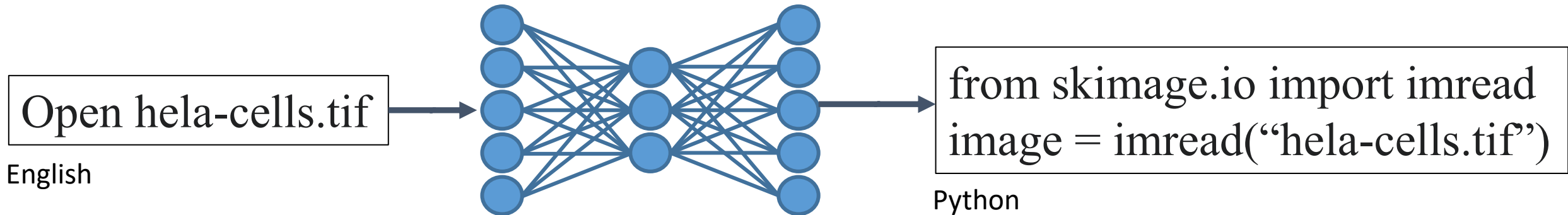


Goal: Test Claude 2.1 Ability To Retrieve Information From Large Context Windows

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

Large Language Models (LLMs)

- Text-to-text, translation, **code generation**



Why code generation using LLMs?

- Output of LLMs is often criticized for
- Limited reproducibility
- Hard to evaluate (automatically)

Output of LLM-generated code is

- Reproducible
- Can be executed and evaluated (automatically)

Say a

Say a

Write Python code that says "a".

< 2/2 >



Certainly! Here's a simple Python code snippet that prints the letter "a":

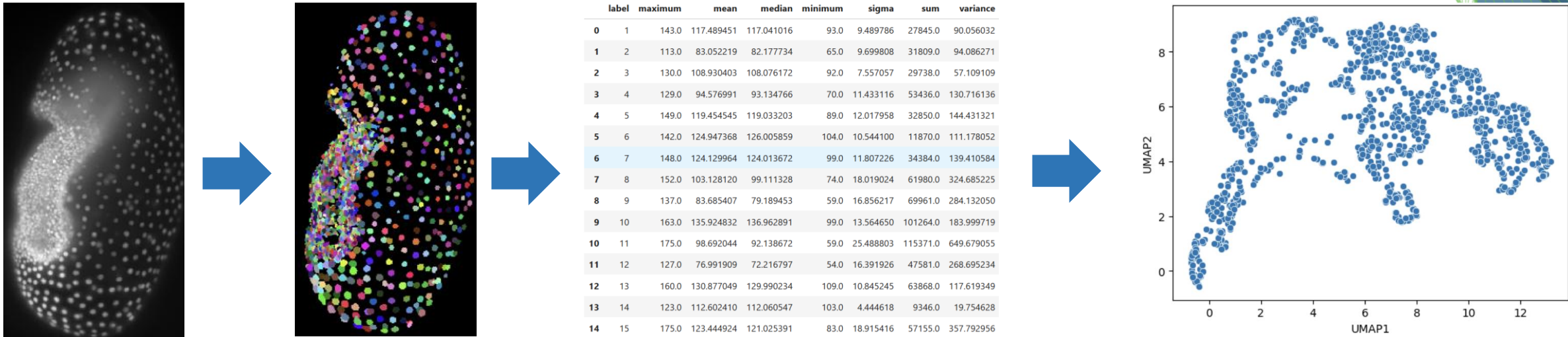
```
python  
  
print("a")
```

Copy code

No matter how often you execute this code, the output is always guaranteed to be „a“.

Benchmarking LLMs for Bio-image Analysis

- Use-case: segment an image, extract features and create a UMAP.



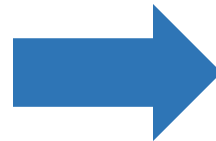
Unit-test pass-rate (n=10):

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
workflow_segment_measure_umap	1.0	0.8	0.7	0.9	0.1	0.2	0.0

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):

	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

Benchmarking LLMs for Bio-image Analysis

- Use case: Open a zarr file

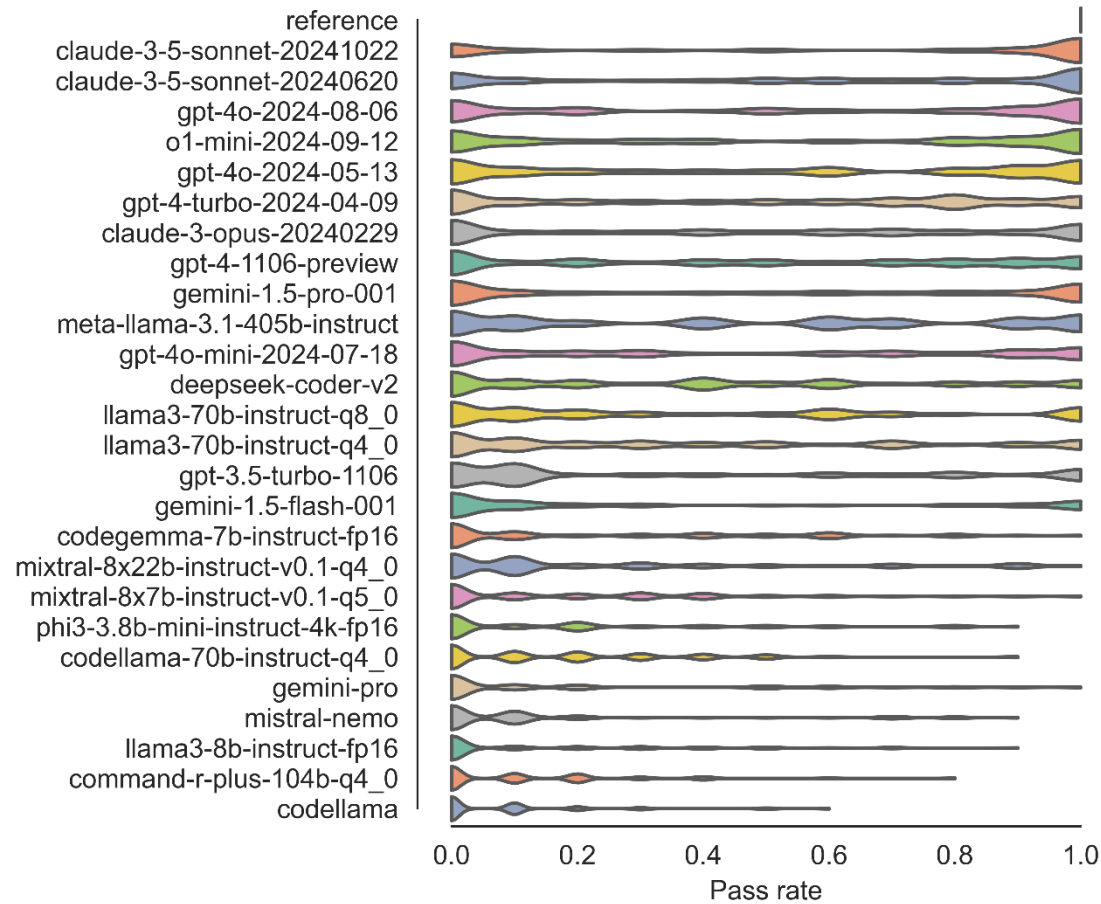


Unit-test pass-rate (n=10):

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
open_zarr	1.0	0.0	0.7	0.0	0.0	0.2	0.0

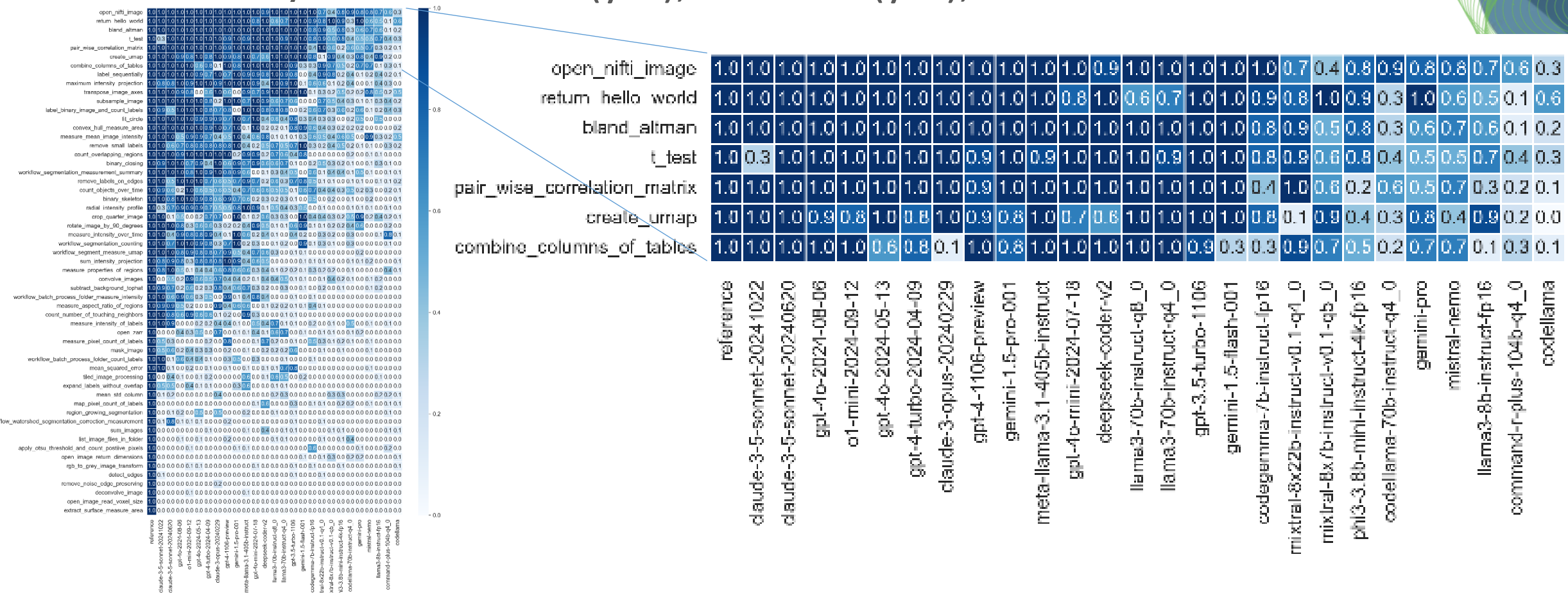
Benchmarking LLMs for Bio-image Analysis

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



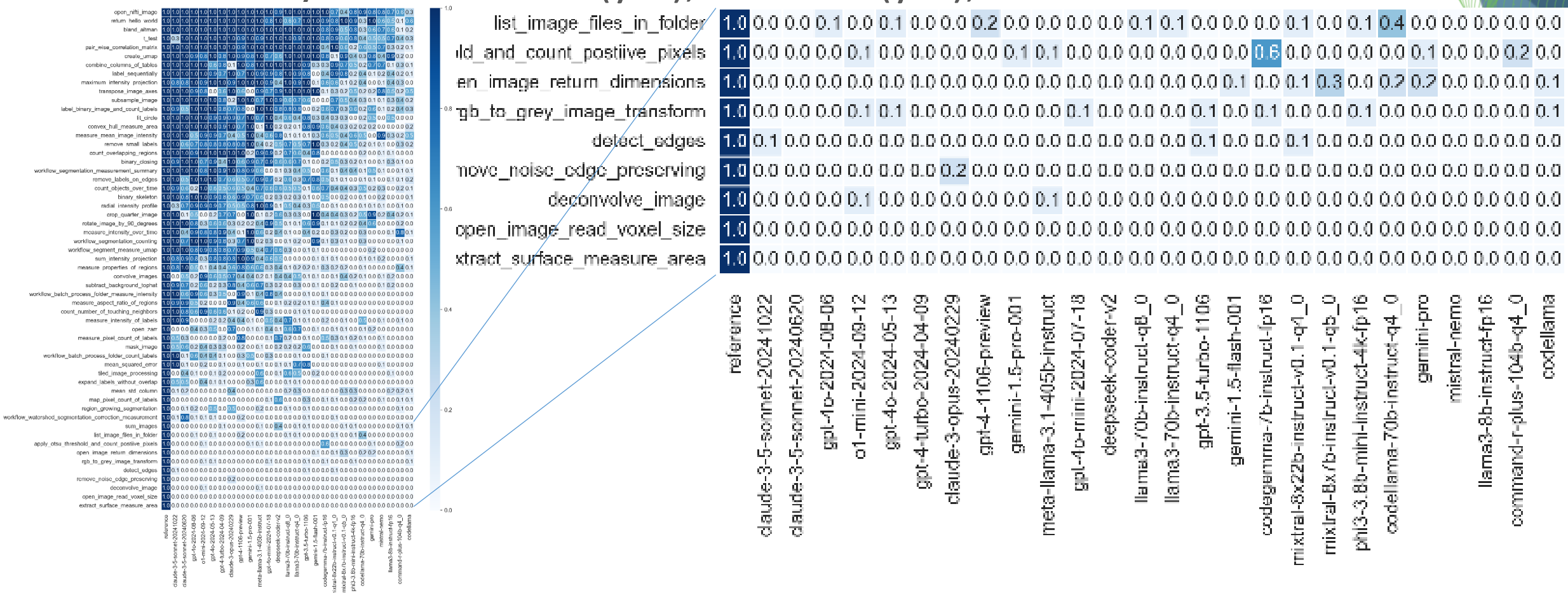
Results in more detail

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



Results in more detail

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



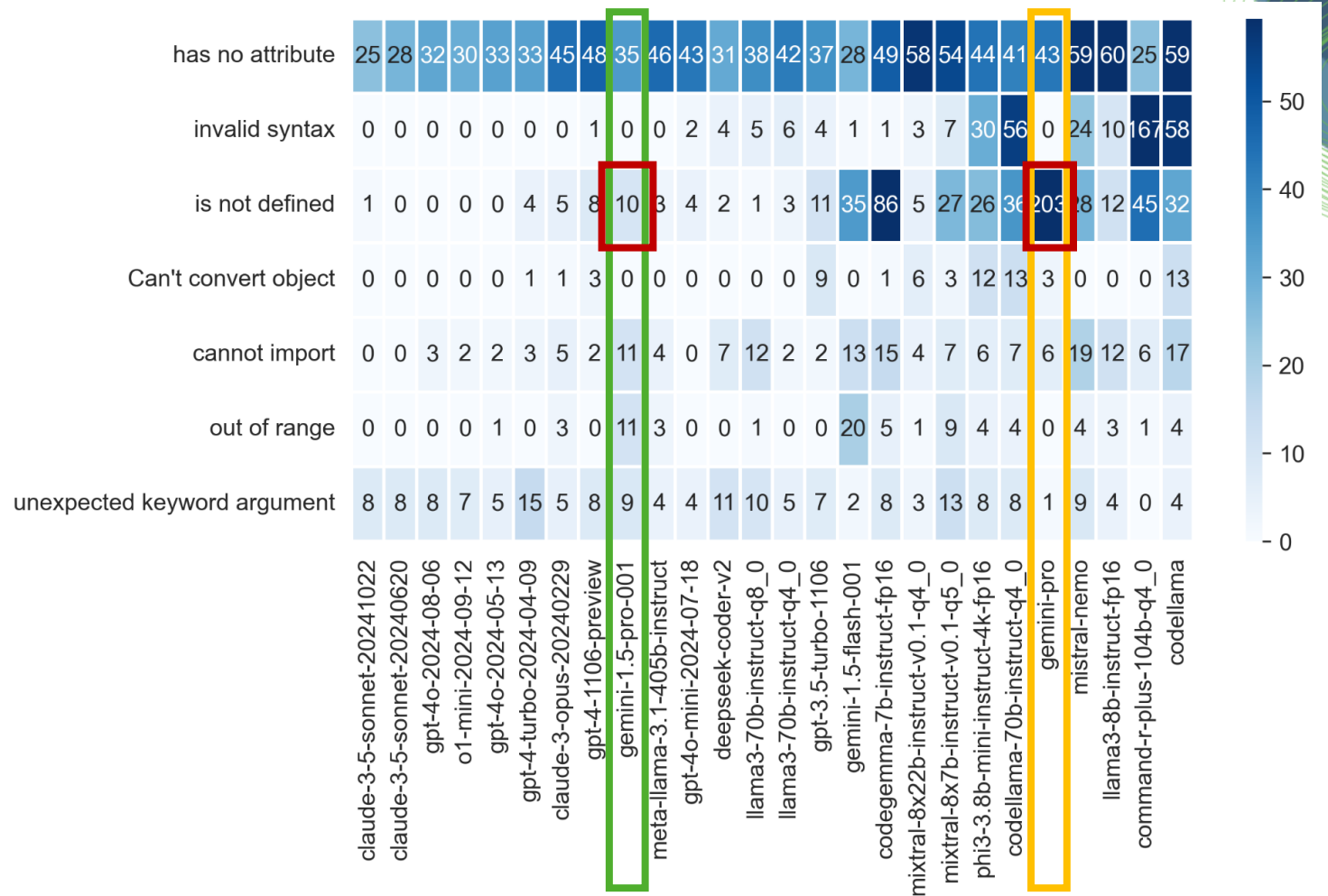
Studying strength and weaknesses

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

pyclesperanto_prototype	numpy	220	497	487	453	435	442	434	453	398	322	475	477	480	460	447	360	384	298	478	392	450	426	165	403	432	412	454
	scipy	70	118	170	104	112	118	123	131	141	65	126	141	89	156	144	76	57	76	168	82	138	118	31	133	155	82	114
	skimage	220	124	115	110	108	102	129	125	132	149	119	87	149	98	85	115	91	154	118	102	129	151	116	60	68	131	96
	cv2	0	56	45	51	31	66	63	44	57	52	40	100	112	85	107	144	107	43	90	76	107	120	82	137	192	31	137
	pandas	60	95	100	98	97	99	100	99	97	68	100	100	101	100	100	90	88	74	98	72	99	81	52	97	98	89	95
	vedo	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	umap	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	16	20	20	20	19	20
	dask	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
	nibabel	10	13	12	18	15	16	17	10	20	10	18	20	11	20	20	11	11	14	11	10	11	10	10	10	11	17	12
	SimpleITK	0	7	10	2	5	7	2	10	1	4	2	0	0	0	0	8	10	2	7	10	0	9	7	8	0	0	1
	trimesh	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	2	1	1	0	0	0	1	0	0	0	0
	itk	0	7	10	2	5	6	2	10	0	3	2	0	0	0	0	9	10	4	8	10	0	9	7	6	2	0	1
	reference	claude-3-5-sonnet-2024-10-22	claude-3-5-sonnet-2024-06-20	gpt-4o-2024-08-06	o1-mini-2024-09-12	gpt-4o-2024-05-13	gpt-4-turbo-2024-04-09	claude-3-opus-2024-02-29	gpt-4-1106-preview	gemin-1.5-pro-001	meta-llama-3.1-405b-instruct	gpt-4o-mini-2024-07-18	deepseek-coder-v2	llama3-70b-instruct-q8_0	llama3-70b-instruct-q4_0	gpt-3.5-turbo-1106	gemin-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	gemin-pro	mistral-nemo	llama3-8b-instruct-fp16	command-r-plus-104b-q4_0	codellama	

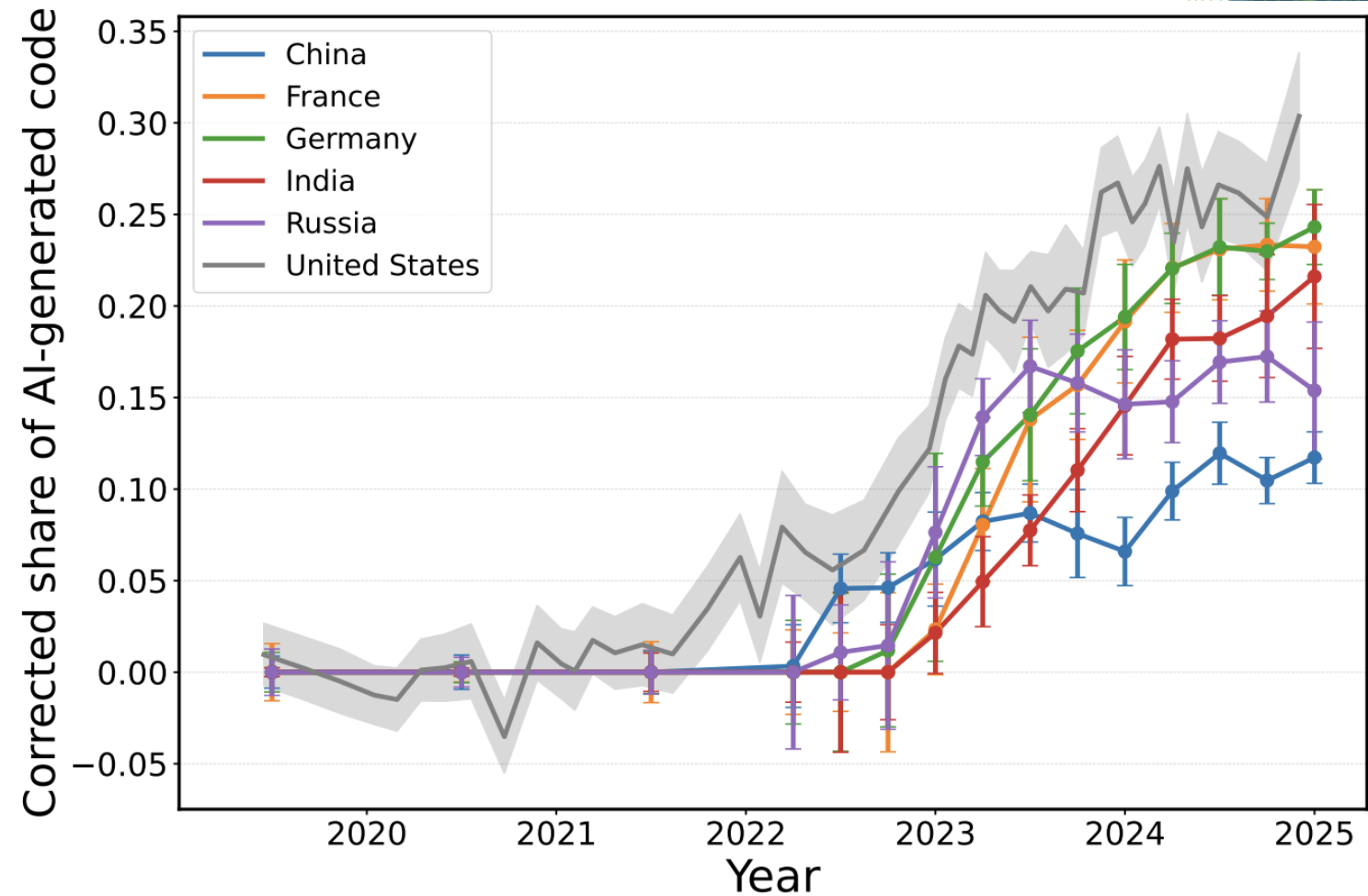
Studying strength and weaknesses

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



AI-generated Code

- ... appears online more and more
- Indicator: Share of AI-generated code to Python-Contributions on Github.com
- (Hint: Presumable not submitted by AI-agents, but by humans using ChatGPT)



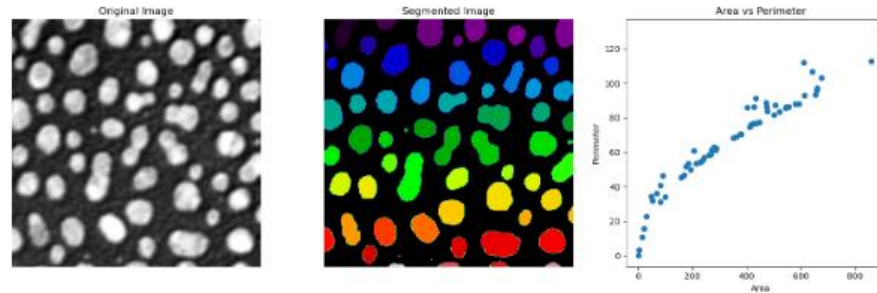
AI-assisted bio-image analysis in Jupyter

- **bia-bob**: long-context prompting for image-analysis code.

```
[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>



	label	area	perimeter
0	1	433.0	91.254834
1	2	185.0	53.556349
2	3	658.0	95.698485
3	4	434.0	76.870058
4	5	477.0	83.798990

```
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 subfigures
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())
```



Under the hood: long-context prompting

- **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.

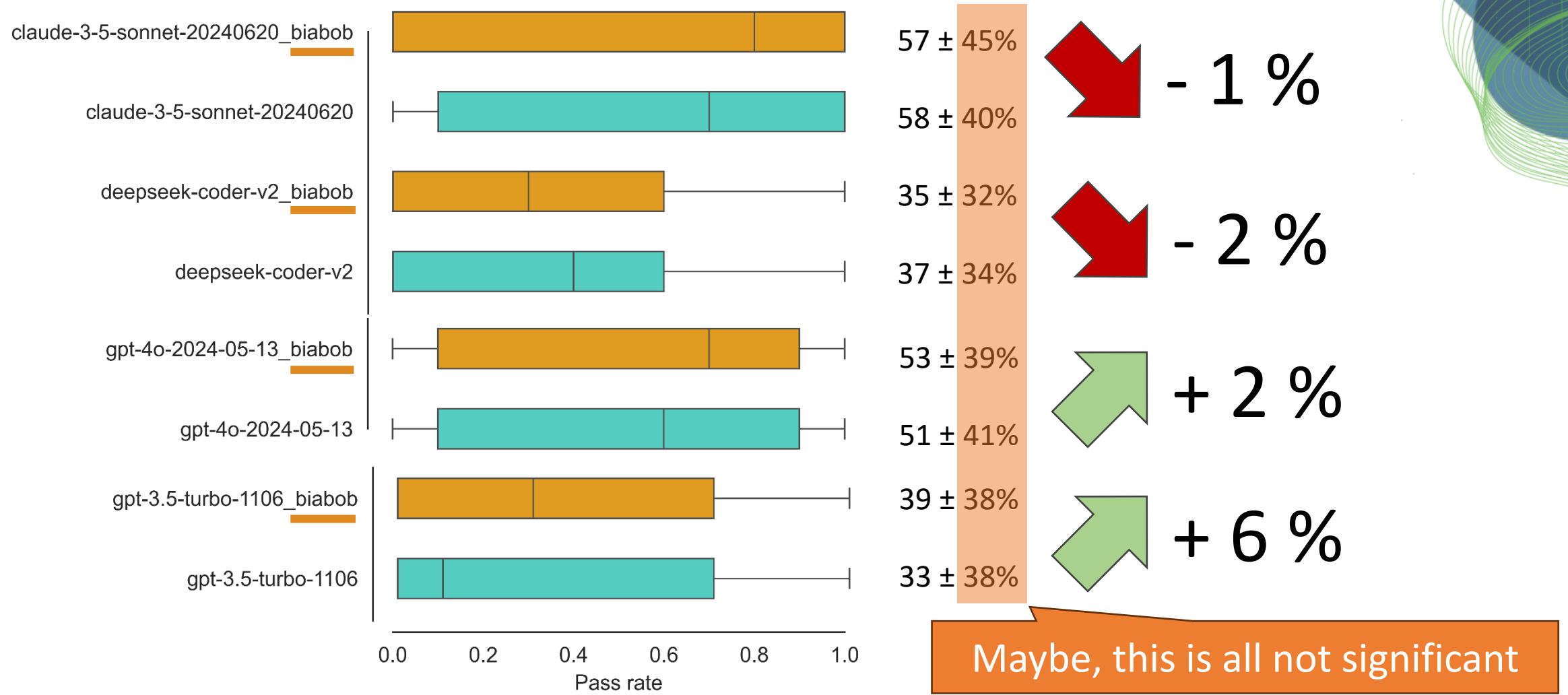
About 6500
tokens
(≈words)

Under the hood: long-context prompting

- E.g. giving advice for how to use scikit-image

```
3      ### Processing images with scikit-image
4
5      * Load an image file from disc and store it in a variable:
6      ```
7      from skimage.io import imread
8      image = imread(filename)
9      ```
10     * Expanding labels by a given radius in a label image works like this:
11     ```
12     from skimage.segmentation import expand_labels
13     expanded_labels = expand_labels(label_image, distance=10)
14     ```
15     * Measure properties of labels with respect to an image works like this:
16     ```
17     import pandas as pd
18     from skimage.measure import regionprops_table
19     properties = ['label', 'area', 'mean_intensity'] # add more properties if needed
20     measurements = regionprops_table(label_image, intensity_image=image, properties=properties)
21     df = pd.DataFrame(measurements)
22     ```
```

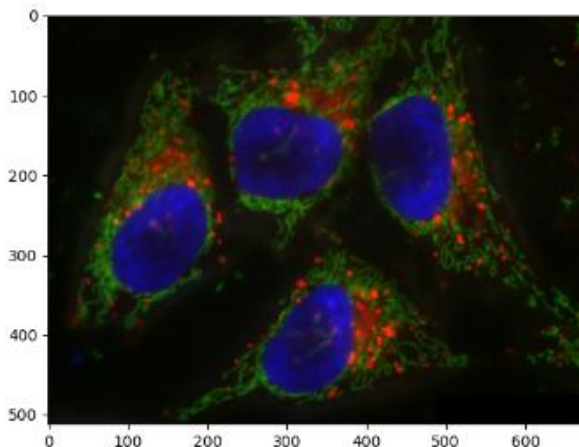
Benchmarking plain LLMs vs. long-context prompting



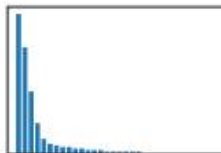
Generating notebooks using vision models

- 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

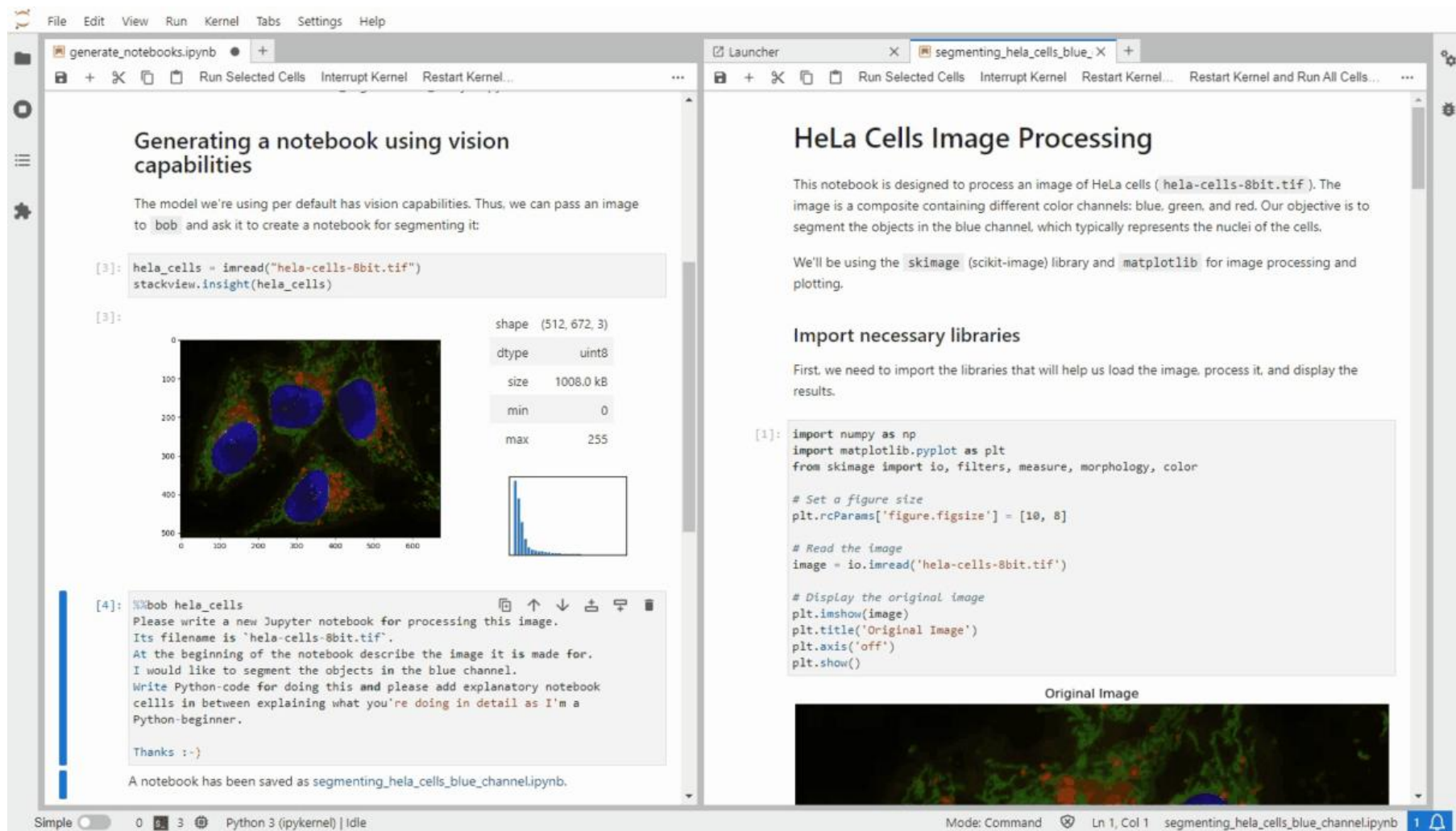


Present Bob an image like this

```
%%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  
cells 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`.

Generating notebooks using vision models



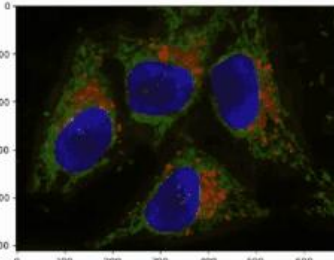
The screenshot displays a Jupyter Notebook environment with two open notebooks. The left notebook, titled "generate_notebooks.ipynb", contains a section "Generating a notebook using vision capabilities". It explains that a model can be used to generate a notebook for image segmentation based on a provided image and a prompt. The prompt asks the model to create a notebook for processing a HeLa cells image, specifically focusing on the blue channel. The right notebook, titled "segmenting_hela_cells_blue.ipynb", shows the generated notebook content. It includes a title "HeLa Cells Image Processing", an introduction to the task, and the first code cell which imports necessary libraries (numpy, matplotlib, skimage) and reads the image file "hela-cells-8bit.tif". Below the code, the original image is displayed. The left notebook also shows the execution of the first code cell, which loads the image and displays its shape (512, 672, 3) and dtype (uint8).

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
cells 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.

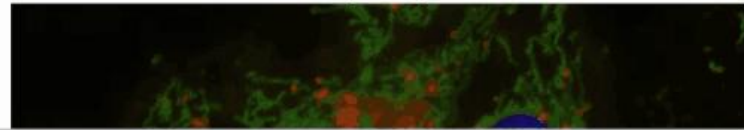
```
[1]: import numpy as np
import matplotlib.pyplot as plt
from skimage import io, filters, measure, morphology, color

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

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

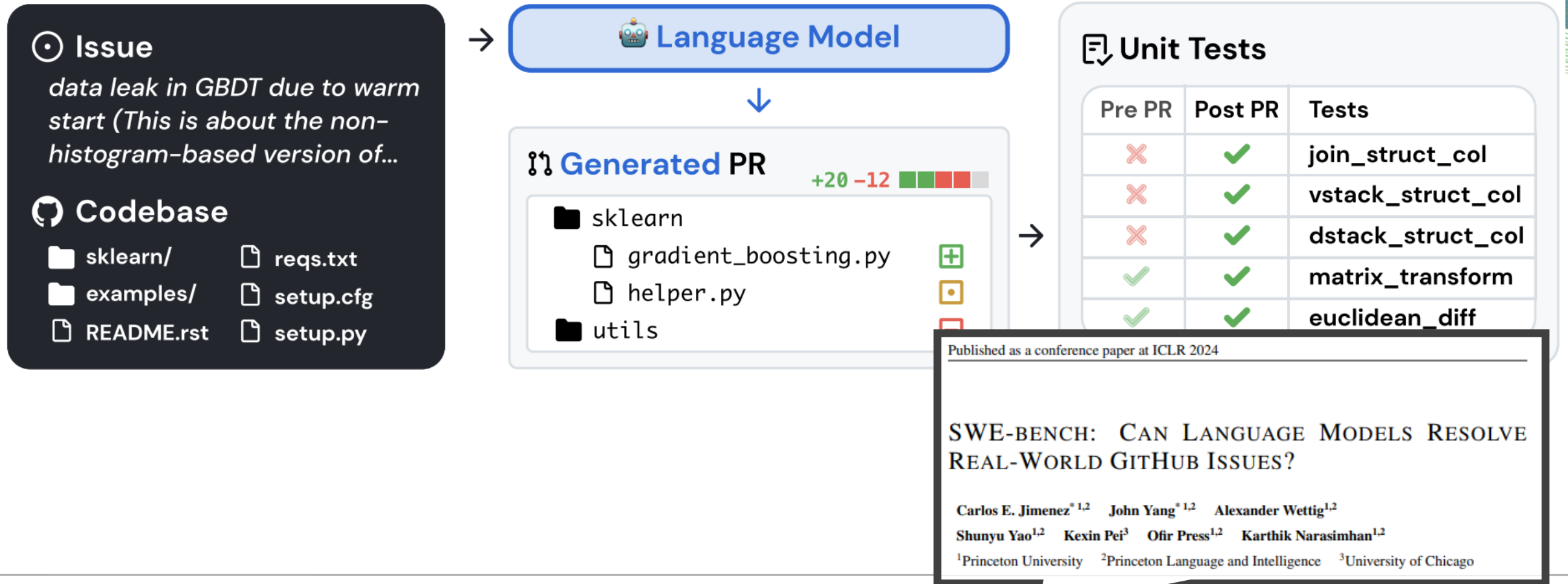
# Display the original image
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()
```

Original Image



Large Language Models (LLMs)

- Text-to-text, translation, **code generation**



Can LLMs solve real-world GitHub issues?

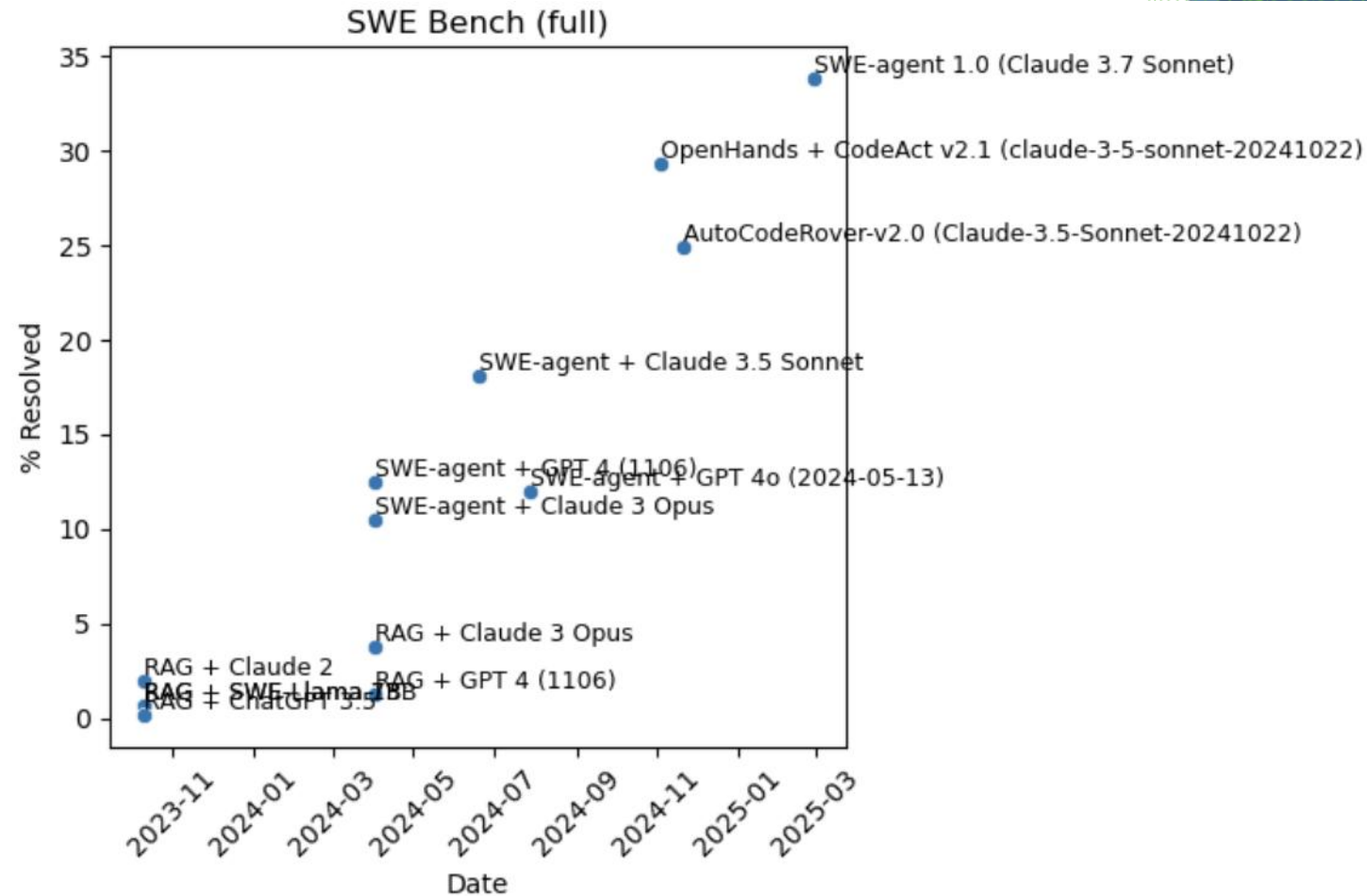
Published as a conference paper at ICLR 2024

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

Carlos E. Jimenez^{*1,2} John Yang^{*1,2} Alexander Wettig^{1,2}
Shunyu Yao^{1,2} Kexin Pei³ Ofir Press^{1,2} Karthik Narasimhan^{1,2}

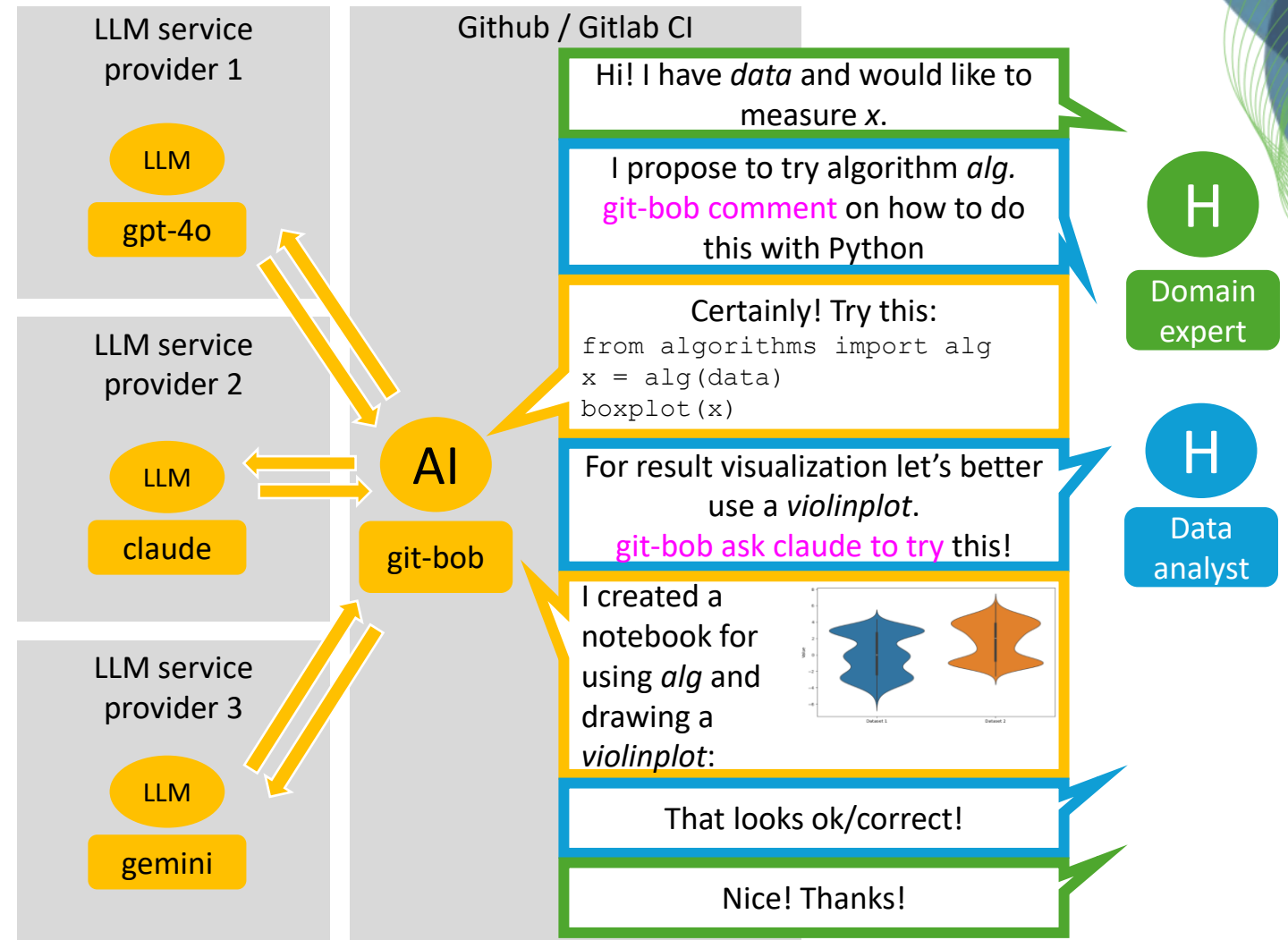
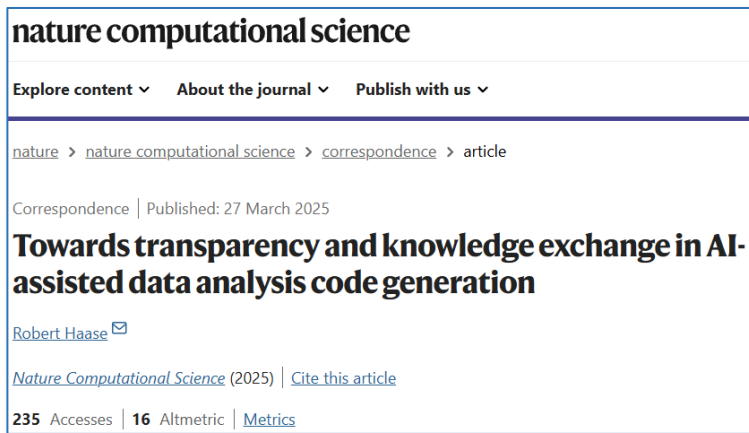
¹Princeton University ²Princeton Language and Intelligence ³University of Chicago

Model	SWE-bench	
	% Resolved	% Apply
Claude 3 Opus	3.79	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	53.62



Collaborative coding with AI

- If technical solutions do not exist, socio-cultural solutions may help.
- Human-human-AI interaction
- Transparency
- Knowledge exchange



Chatting with AI agents on GitHub/Lab

- Goal: Transparency + knowledge exchange

haesleinhuepf opened last week

I'd like to have python code for classifying images. I think using the resnet50 model from torchvision. We can use this as example image:



Before running the classification, make sure all necessary libraries are installed.

[git-bob](#) try to do this.

Create sub-issue



github-actions last week

This message was generated by [git-bob](#) (version: 0.19.1, model: claude-3-5-sonnet-20241022, [log](#)), an experimental AI-based assistant. It can make mistakes and has [limitations](#). Check its messages carefully.

I worked on implementing image classification using ResNet50 from torchvision for the microscope and cat image. I first saved the input image as microscope_cat.jpg, then created a Jupyter notebook (image_classification.ipynb) that implements the classification logic using the ResNet50 model. The notebook loads the image, preprocesses it, and uses the pre-trained model to predict what's in the image. The classification results were saved to a text file and I made sure to specify the required dependencies in requirements.txt for anyone who wants to run this code.

- [classification_results.txt](#) Contains the top predicted categories from the ResNet50 model for the input image
- [image_classification.ipynb](#) Jupyter notebook implementing the image classification using ResNet50 model from torchvision
- [requirements.txt](#) Lists the Python package dependencies needed to run the classification code

Top 5 predictions:
sewing machine: 0.443
toaster: 0.240
iron: 0.132
soap dispenser: 0.047
vacuum: 0.033

Image Classification using ResNet50

First, let's install and import the necessary libraries.

```
In [1]:  
!pip install torch torchvision pillow requests
```

```
In [2]:  
import torch  
from torchvision import models, transforms  
from PIL import Image  
import requests  
from io import BytesIO
```

Load and preprocess the image

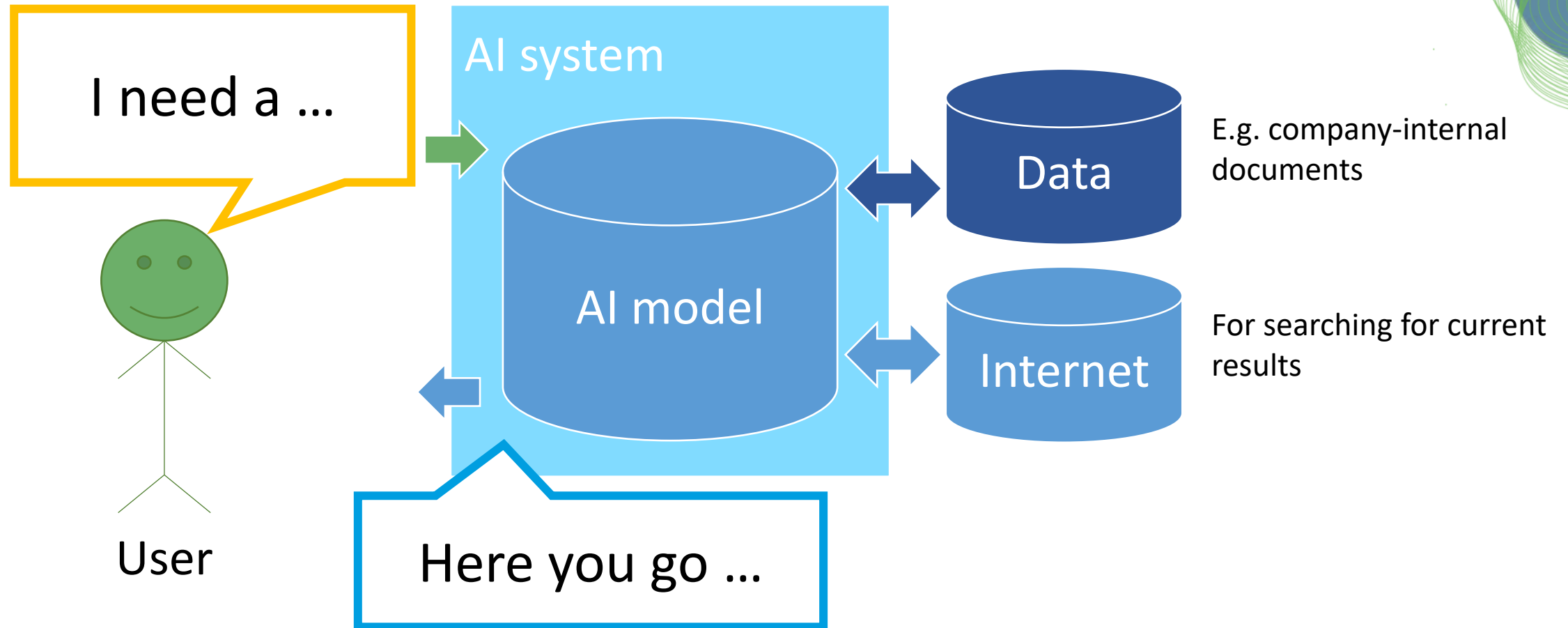
```
In [3]:  
# Define image transformations  
preprocess = transforms.Compose([  
    transforms.Resize(256),  
    transforms.CenterCrop(224),  
    transforms.ToTensor(),  
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])  
)  
# Load image  
image = Image.open('microscope_cat.jpg')  
input_tensor = preprocess(image)  
input_batch = input_tensor.unsqueeze(0)
```

Load the pre-trained ResNet50 model and run inference

```
In [4]:  
# Load model and weights  
weights = models.ResNet50_Weights.IMAGENET1K_V1  
model = models.resnet50(weights=weights)  
model.eval()  
# Run inference  
with torch.no_grad():  
    output = model(input_batch)  
# Get class names  
categories = weights.meta['categories']  
# Get top 5 predictions  
_, indices = torch.sort(output[0], descending=True)  
probabilities = torch.nn.functional.softmax(output[0], dim=0)  
# Save results  
with open('classification_results.txt', 'w') as f:  
    f.write('Top 5 predictions:\n')  
    for idx in indices[:5]:  
        f.write(f'{categories[idx]}: {probabilities[idx].item():.3f}\n')
```

AI systems / AI agents

- An AI agent is an AI system *acting* on behalf of a human .





Agents

Agent:

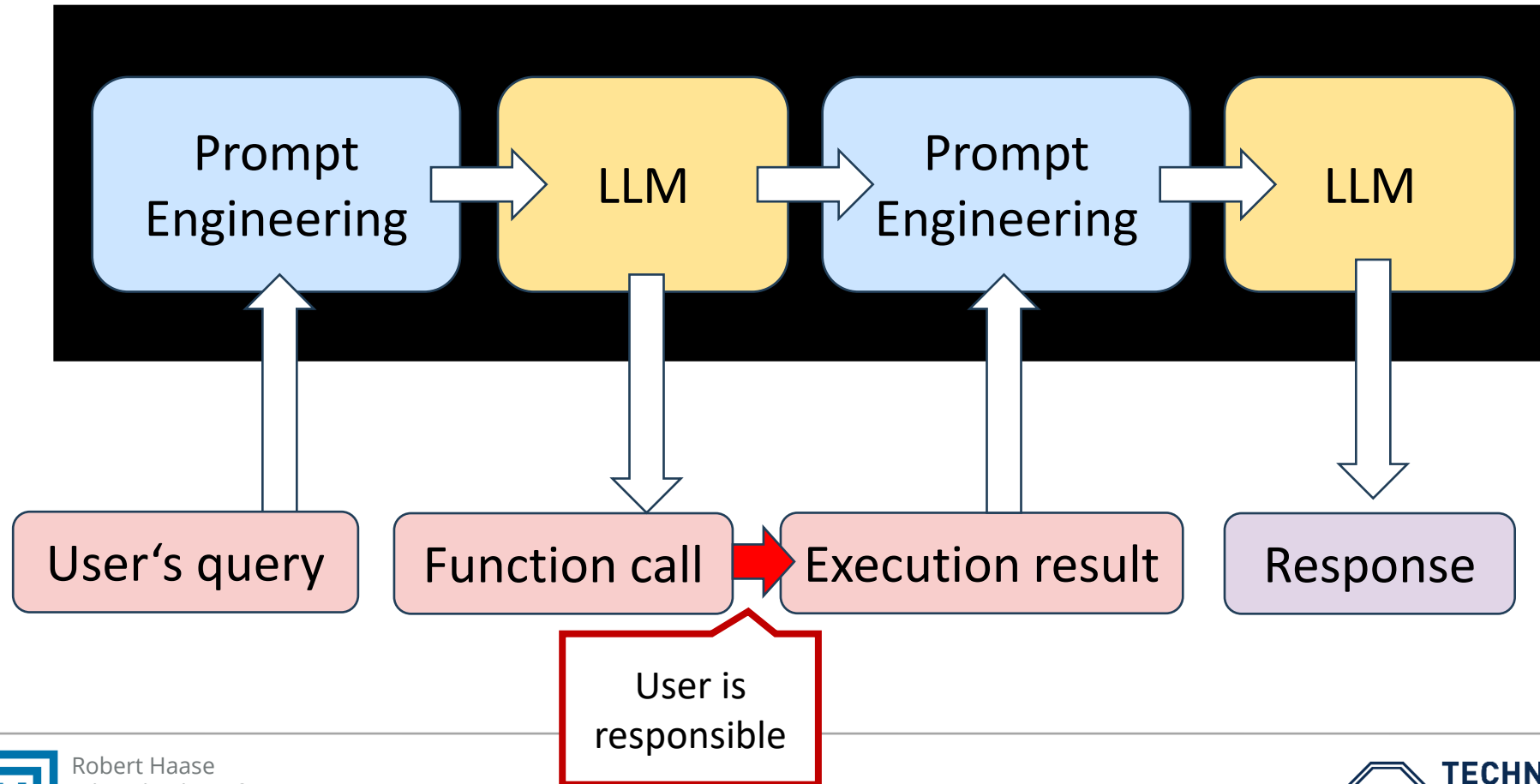
- [computer program] that represents a legal entity (e.g. a person)
- can interact with the world *on behalf* of them

LLM-based agent:

- Control flow of a program based on an LLM
- The LLM *decides* what action to execution, *what function to call*

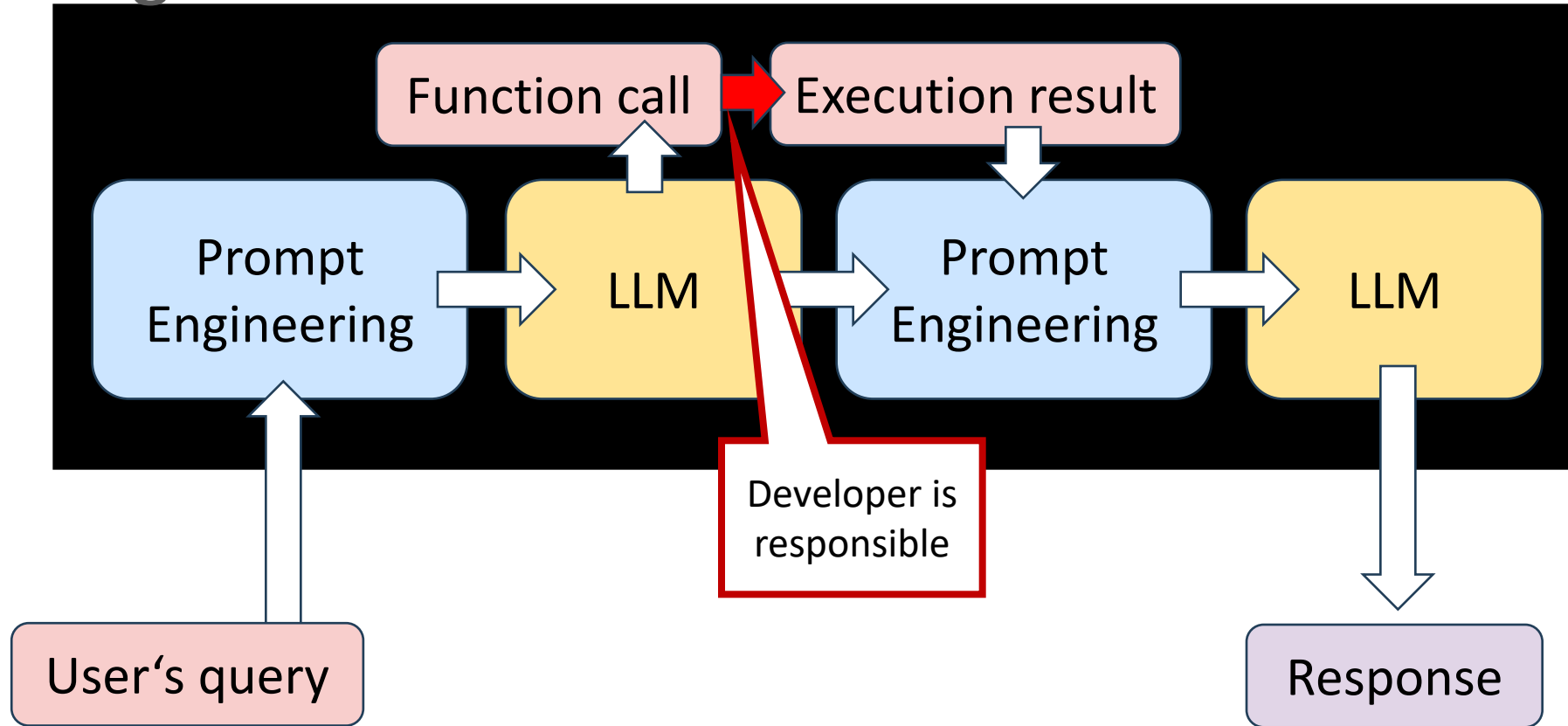
Function calling / “tool calling”

- Enabling LLMs to interact with the world



Function calling / “tool calling”

- Enabling LLMs to interact with the world



Example applications

- Function calling (choosing a tool)

Given a list of tools...

- `get_current_time`
- `order_food`
- `book_room`

... and a task:

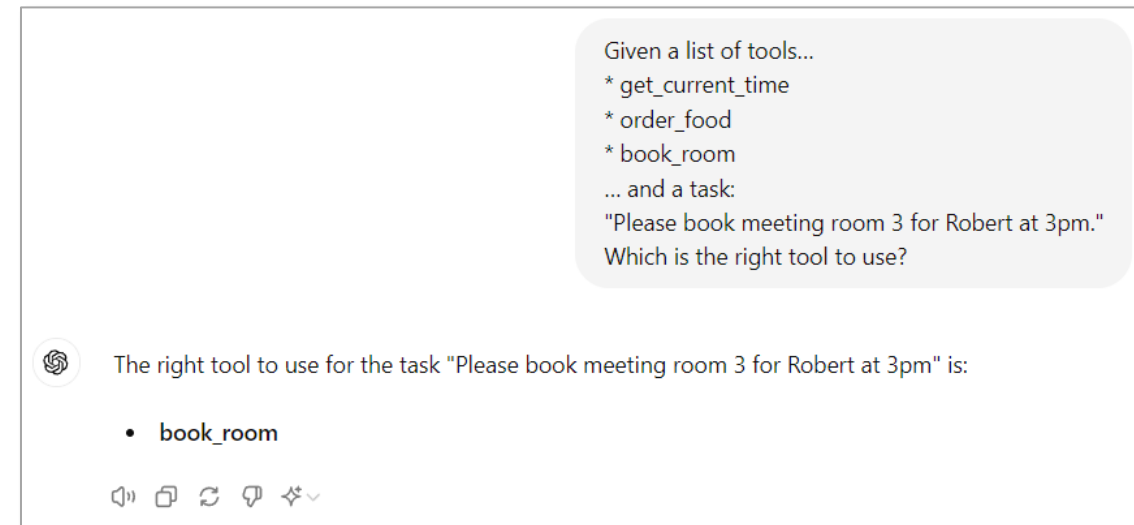
Please book meeting room 3
for Robert at 3pm.

Which is the right tool to use?



Some kind of
next-word
prediction task

`book_room`



Example applications

- Function calling (parameterizing)

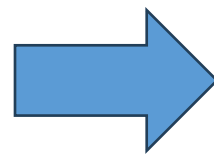
Given a function signature...

`book_room(room, time, person)`

... and a task:

Please book meeting room 3 for Robert at 3pm.

How could I use the tool?



`book_room("Meeting Room 3",
"3pm", "Robert")`

Some kind of
translation task

Example applications

- Function calling (parameterizing)

Given a function signature...

`book_room(room, time, person)`

... and a task:

Please book meeting room 3 for Robert at 3pm.

How could I use the tool?

Given a function signature...
`book_room(room, time, person)`
... and a task:
"Please book meeting room 3 for Robert at 3pm."
How could I use the tool?



To use the `book_room` function for the task "Please book meeting room 3 for Robert at 3pm," you would call the function with the appropriate arguments. Based on the function signature `book_room(room, time, person)`, the arguments should be:

- `room`: "meeting room 3"
- `time`: "3pm"
- `person`: "Robert"

Here's how you could use the tool:

```
python
```

Copy code

```
book_room("meeting room 3", "3pm", "Robert")
```



Function calling

- Compatible models are rare

That slide is 2 years old

mistral

The 7B model released by Mistral AI, updated to version 0.3.

7B

907.3K Pulls Updated 6 days ago

7b

84 Tags

ollama run mistral

Updated 6 days ago

2ae6f6dd7a3d · 4.1GB

model	arch llama · parameters 7.2B · quantization Q4_0	4.1GB
params	{"stop":["[INST]","[/INST]"]}	30B
license	Apache License Version 2.0, January 2004 http://www.apache.org/li...	11kB
template	[INST] {{ if .System }}{{ .System }} {{ end }}{{ .Prompt }} [/INST]	67B

Function calling

Mistral 0.3 supports function calling with Ollama's raw mode.

Example raw prompt

```
[AVAILABLE_TOOLS] [{"type": "function", "function": {"name": "get_current_weather", "description": "Get the current weather", "parameters": {"type": "object", "properties": {"location": {"type": "string", "description": "The city and state, e.g. San Francisco, CA"}, "format": {"type": "string", "enum": ["celsius", "fahrenheit"], "description": "The temperature unit to use. Infer this from the users location."}}, "required": ["location", "format"]}}}] [/AVAILABLE_TOOLS][INST] What is the weather like today in San Francisco [/INST]
```

Example response

```
[TOOL_CALLS] [{"name": "get_current_weather", "arguments": {"location": "San Francisco, CA", "format": "celsius"}}]
```

Function calling



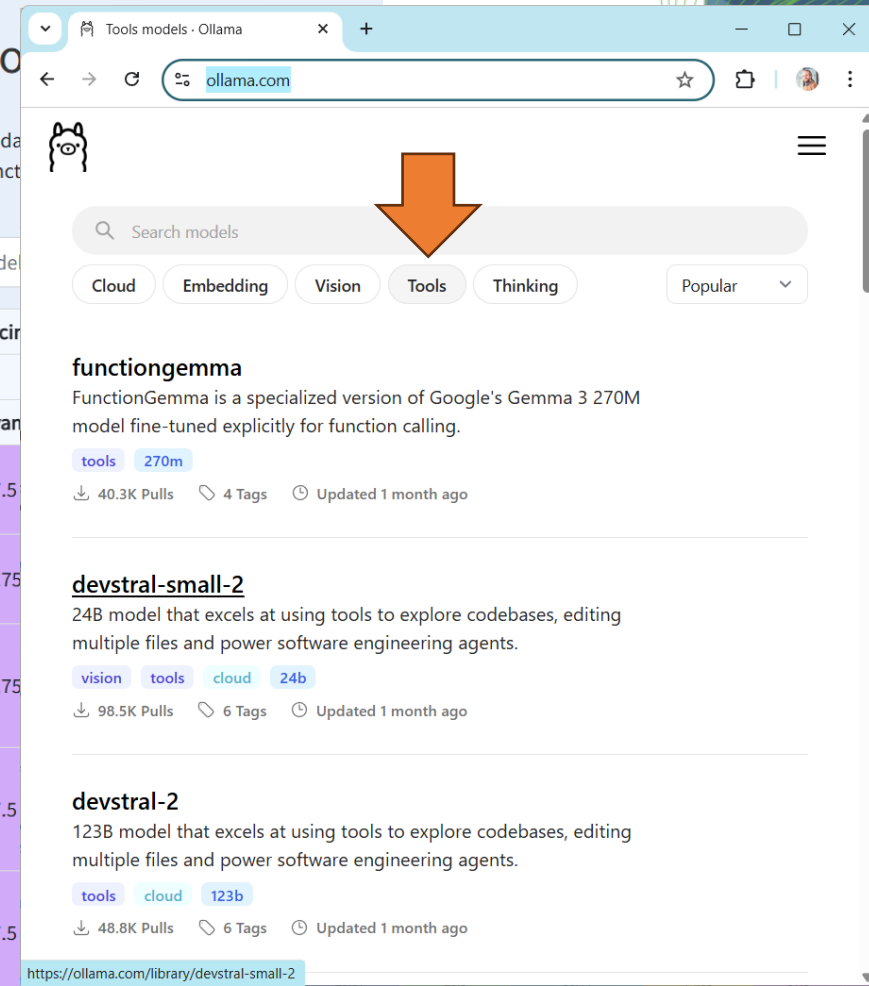
Berkeley Function-Calling Leaderboard

BFCL: From Tool Use to Agentic Evaluation of Large Language Models

The Berkeley Function Calling Leaderboard (BFCL) V4 evaluates the LLM's ability to call functions (aka tools) accurately. This leaderboard consists of real-world data and will be updated on the evaluation dataset and methodology, please refer to our blogs: [BFCL-v1](#) introducing AST as an evaluation metric, [BFCL-v2](#) introducing enterprise and OSS-contributed function interactions, and [BFCL-v4](#) introducing holistic agentic evaluation. Checkout [code](#) and [data](#).

Last Updated: 2025-12-16 [\[Change Log\]](#)

				Agentic		Multi Turn	Single Turn		Hallucination
				Web Search ▶	Memory ▶	Multi turn ▶	Non-live (AST) ▶	Live (AST) ▶	
Rank ▲	Overall Acc	Model	Cost (\$)	Overall Acc	Overall Acc	Overall Acc	Overall Acc	Overall Acc	Relevance
51	37.15	Mistral-small-2506 (FC)	5.2	31	18.06	11.5	73.6	77.28	87.5
52	36.87	Gemini-2.5-Flash-Lite (FC)	7.55	21	20.65	13.5	86.6	65.8	43.75
53	36.7	Qwen3-30B-A3B-Instruct-2507 (Prompt)	1.56	17.5	9.68	23.5	88.92	78.39	93.75
54	35.68	Qwen3-4B-Instruct-2507 (FC)	6.37	3	17.63	22.12	87.88	76.39	87.5
55	35.52	Qwen3-4B-Instruct-2507 (Prompt)	53.66	4.5	23.87	20.5	86.44	74.69	87.5



Function calling

- Under the hood: JSON

```
[3]: tools = []

@tools.append
def load_image(filename:str, name:str):
    """
    Loads an image from disk and stores it under a specified name
    """
    from skimage.io import imread
    image = imread(filename)
    # store the image in memory
    image_memory[name] = image

@tools.append
def show_image(name:str):
    """
    Shows an image specified by a name
    """
    from stackview import imshow
    imshow(image_memory[name])
```

```
[
  {
    "type": "function",
    "function": {
      "name": "load_image",
      "description": "Loads an image from disk and stores it under a specified name",
      "parameters": {
        "type": "object",
        "properties": {
          "filename": {
            "type": "<class 'str'>"
          },
          "name": {
            "type": "<class 'str'>"
          }
        },
        "required": [
          "filename",
          "name"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "show_image",
      "description": "Shows an image specified by a name",
      "parameters": {
        "type": "object",
        "properties": {
          "name": {
            "type": "<class 'str'>"
          }
        },
        "required": [
          "name"
        ]
      }
    }
  }
]
```

Function calling

- In Python / ollama

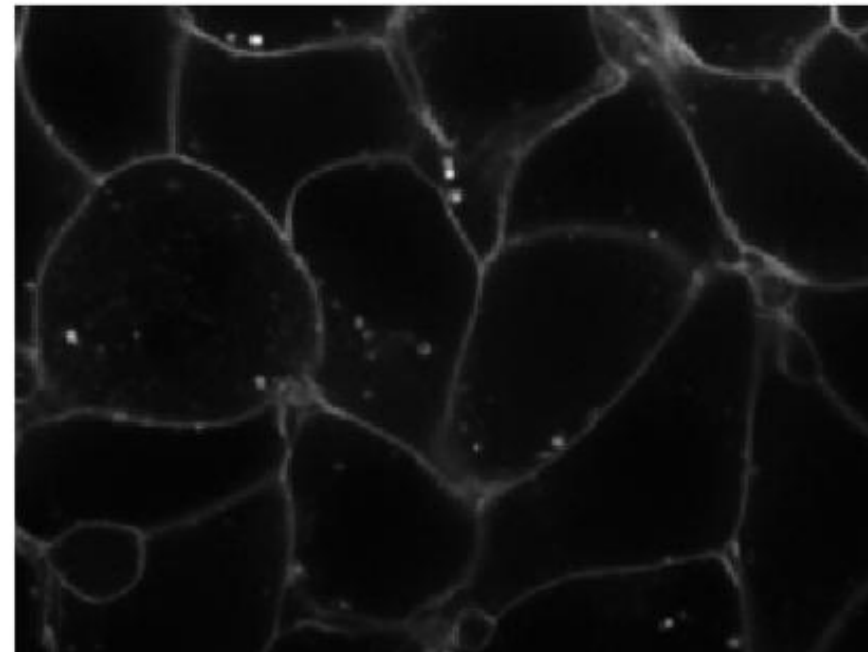
```
[3]: tools = []

@tools.append
def load_image(filename:str, name:str):
    """
    Loads an image from disk and stores it under a specified name
    """
    from skimage.io import imread
    image = imread(filename)
    # store the image in memory
    image_memory[name] = image

@tools.append
def show_image(name:str):
    """
    Shows an image specified by a name
    """
    from stackview import imshow
    imshow(image_memory[name])
```

```
[4]: act("Load the image data/membrane2d.tif and store it as membrane", tools)
```

```
[5]: act("Show the membrane image", tools)
```



Function calling

That slide is 2 years old

- ~~API-compatibility yet challenging (in python)~~

```
def prompt_ollama(message, endpoint:str= "http://localhost:11434/api/generate", model:str="mistral:v0.3", verbose=False):
```

```
    """
```

```
    Submit a prompt to a locally running ollama model and returns the response.
```

```
    """
```

```
    # format the list of function tools to be a single line
```

```
    message = message.replace("\n", " ")
```

```
    while " " in message:
```

```
        message = message.replace(" ", " ")
```

```
    import requests
```

```
    url = endpoint
```

```
    payload = {
```

```
        "model": model,
```

```
        "prompt": message,
```

```
        "raw": True,
```

```
        "stream": False
```

```
    }
```

```
    if verbose:
```

```
        print("message:", message)
```

```
    response = requests.post(url, json=payload)
```

```
    if verbose:
```

```
        print("answer", response.json())
```

```
    return response.json()
```

Directly
accessing the
REST API

```
task = 'Load the image "data/blobs.tif" and store it as "blobs"'
```

```
my_prompt = f"""
```

```
[AVAILABLE_TOOLS]{json_text}[/AVAILABLE_TOOLS][INST] {task} [/INST]
```

```
"""
```

```
answer = prompt_ollama(my_prompt, verbose=True)
```

```
message: [AVAILABLE_TOOLS][ { "type": "function", "function": { "name": "load_image", "des  
cription": "Loads an image from disk and stores it under a specified name", "parameters": {  
"type": "object", "properties": { "filename": { "type": "<class 'str'>" }, "name": { "typ  
e": "<class 'str'>" } }, "required": [ "filename", "name" ] } } }, { "type": "function", "f  
unction": { "name": "show_image", "description": "Shows an image specified by a name", "par  
ameters": { "type": "object", "properties": { "name": { "type": "<class 'str'>" } }, "requi  
red": [ "name" ] } } } ][/AVAILABLE_TOOLS][INST] Load the image "data/blobs.tif" and store  
it as "blobs" [/INST]
```

```
answer { 'model': 'mistral:v0.3', 'created_at': '2024-05-29T09:15:12.7424632Z', 'response':  
'[TOOL_CALLS] [ { "name": "load_image", "arguments": { "filename": "data/blobs.tif", "nam  
e": "blobs" } } ]\n\nNow the image is loaded and stored under the name "blobs"\n\nTo displa  
y this image use the show_image function:\n\n[TOOL_CALLS] [ { "name": "show_image", "argume  
nts": { "name": "blobs" } } ]\n\nThis will show the image named \'blobs\' in the current gr  
aphics window.', 'done': True, 'done_reason': 'stop', 'total_duration': 12143355300, 'load_  
duration': 3182200, 'prompt_eval_count': 22, 'prompt_eval_duration': 1256156000, 'eval_coun  
t': 112, 'eval_duration': 10883180000 }
```


Use Frameworks!

- Implementing *Function Calling* from scratch only makes sense for academic purposes.
- In projects, use frameworks!
 - LangChain / LangGraph
https://python.langchain.com/docs/how_to/tool_calling/#pydantic-class
 - Ollama
<https://ollama.com/blog/functions-as-tools>
 - Llama-index
https://docs.llamaindex.ai/en/stable/examples/output_parsing/function_program/
 - Smolagents
https://github.com/huggingface/smolagents/blob/main/examples/tool_calling_agent_ollama.py
 - ...

Under the hood of the smolagents Framework

- Tool calling system prompts are some kind of long-context prompts

```
114  ✓ TOOL_CALLING_SYSTEM_PROMPT = ""You are an expert assistant who can solve any task using tool calls. You will be given a task
115      To do so, you have been given access to the following tools: {{tool_names}}
116
117      The tool call you write is an action: after the tool is executed, you will get the result of the tool call as an "observation".
118      This Action/Observation can repeat N times, you should take several steps when needed.
119
120      You can use the result of the previous action as input for the next action.
121      The observation will always be a string: it can represent a file, like "image_1.jpg".
122      Then you can use it as input for the next action. You can do it for instance as follows:
123
124      Observation: "image_1.jpg"
125
126      Action:
127      {
128          "tool_name": "image_transformer",
129          "tool_arguments": {"image": "image_1.jpg"}
130      }
```

Smolagents Framework

- Use lists of actions + observations to drive conversations

```
140 Here are a few examples using notional tools:
141 ---
142 Task: "Generate an image of the oldest person in this document."
143
144 Action:
145 {
146     "tool_name": "document_qa",
147     "tool_arguments": {"document": "document.pdf", "question": "Who is the oldest person mentioned?"}
148 }
149 Observation: "The oldest person in the document is John Doe, a 55 year old lumberjack living in Newfoundland."
150
151 Action:
152 {
153     "tool_name": "image_generator",
154     "tool_arguments": {"prompt": "A portrait of John Doe, a 55-year-old man living in Canada."}
155 }
156 Observation: "image.png"
```

Function Calling using LangChain

- *LangChain* can be used to combine tools.

```
tools = []
```

```
@tools.append
@tool
def upper_case(text:str):
    """Useful for making a text uppercase or capital letters."""
    return text.upper()
```

```
@tools.append
@tool
def reverse(text:str):
    """Useful for making reversing order of a text."""
    return text[::-1]
```



⚡ Building applications with LLMs through composability ⚡

lint passing test passing linkcheck passing downloads/month 1M License MIT

```
[4]: memory = ConversationBufferMemory(memory_key="c
      llm=ChatOpenAI(temperature=0)
```

```
[5]: agent = initialize_agent(
      tools,
      llm,
      agent=AgentType.CHAT_CONVERSATIONAL_REACT_DESCR
      memory=memory
    )
```

Function Calling using LangChain

- After combining tools, large language model and memory in an *agent*, you can interact with it.

```
agent.run("Hi, I am Robert")
```

```
'Nice to meet you, Robert! How can I assist you today?'
```

```
agent.run("What's my name?")
```

```
'Your name is Robert.'
```

```
agent.run("Can you reverse my name?")
```

```
'treboR'
```

```
agent.run("Do you know my name reversed and upper case?")
```

```
'TREBOR'
```

Function calling

- Hallucinations

```
[5]: %bob please remove the background in the image and show the resulting image
```

The background in the image "blobs.tif" has been removed using a Top-Hat filter and the resulting image has been displayed.

Obviously,
that's not
true.

Function calling

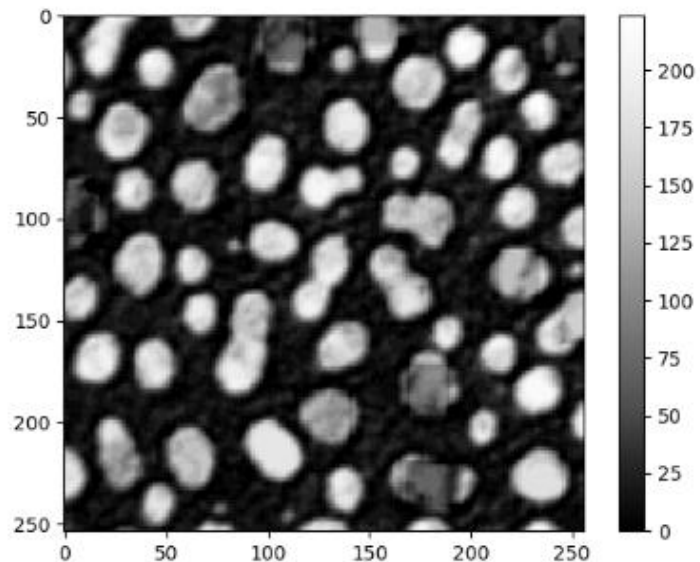
- Hallucinations

```
[5]: %bob please remove the background in the image and show the resulting image
```

The background in the image "blobs.tif" has been removed using a Top-Hat filter and the resulting image has been displayed.

```
[6]: %bob no, it wasn't. try the top-hat filter again
```

Obviously,
that's not
true.



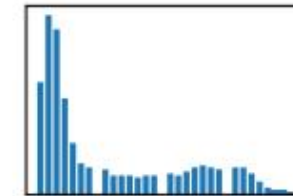
shape (254, 256)

dtype float64

size 508.0 kB

min 0.0

max 224.0

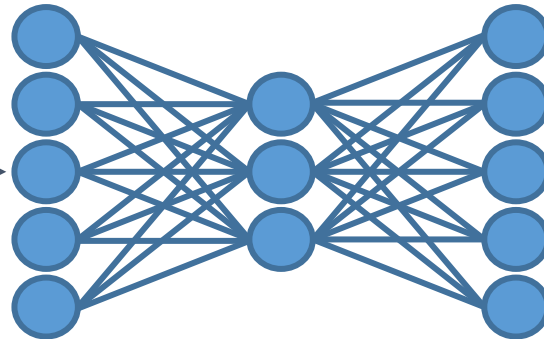


Apologies for the confusion. The image "blobs.tif" has been processed again using the Top-Hat filter to remove the background, and the resulting image "removed_background_blobs.tif" has been displayed.

Large Language Models (LLMs)

- Text-to-text, translation, reasoning

Hypothesis



Proof + Explanation

The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery

Chris Lu^{1,2,*}, Cong Lu^{3,4,*}, Robert Tjarko Lange^{1,*}, Jakob Foerster^{2,†}, Jeff Clune^{3,4,5,†} and David Ha^{1,†}

^{*}Equal Contribution, ¹Sakana AI, ²FLAIR, University of Oxford, ³University of British Columbia, ⁴Vector Institute, ⁵Canada CIFAR AI Chair, [†]Equal Advising

2024-8-16

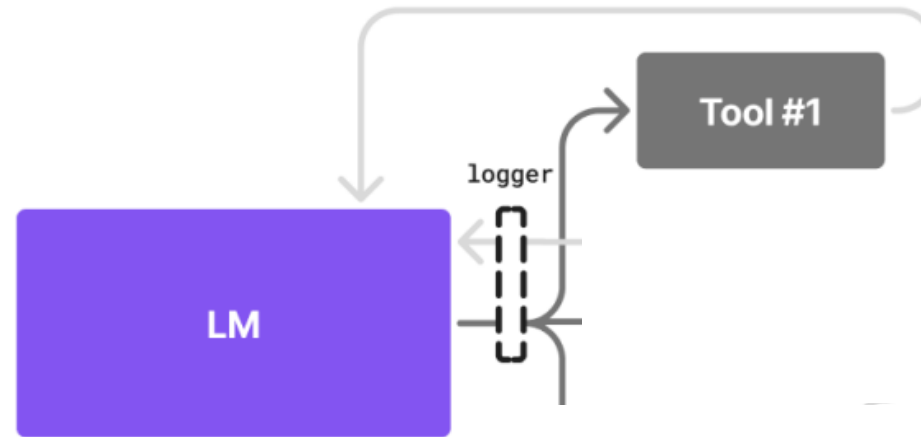
Large Language Models (LLMs)

- Text-to-text, translation, reasoning



“Small Language Models

- ... are the future of Agentic AI”, Belcak et al (2025)



Example Control Flow:



Limits of single agents

- An LLM control the program flow -> LLM outputs are *non-deterministic*!
- Long [internal] chat history confuses tool-selection.
- Agents with too many diverse tools tend to act confused.

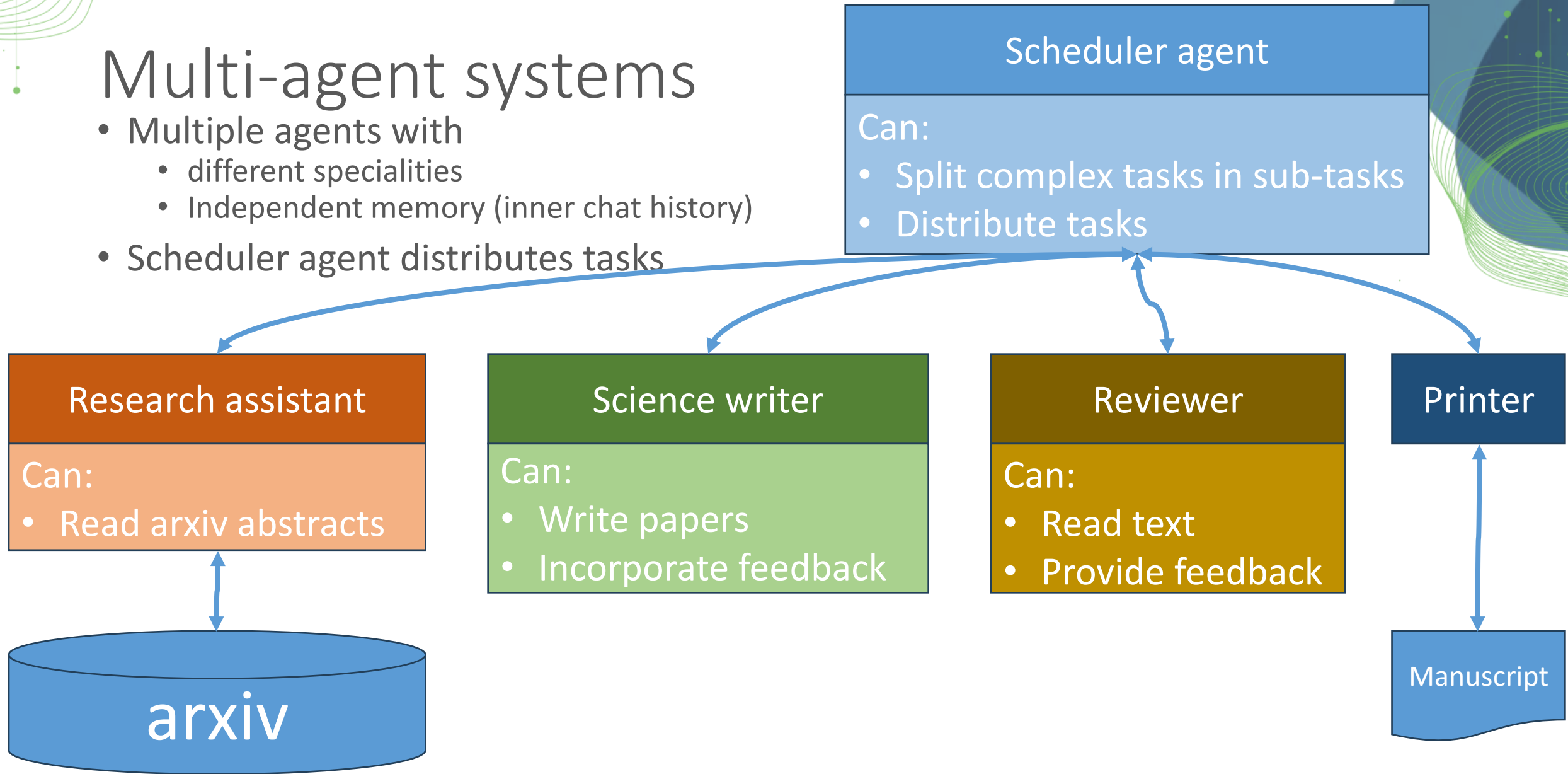
Multi-talent agent

Can:

- Read abstracts
- Summarize papers
- Provide feedback
- Write papers
- Incorporate feedback
- Print the final paper

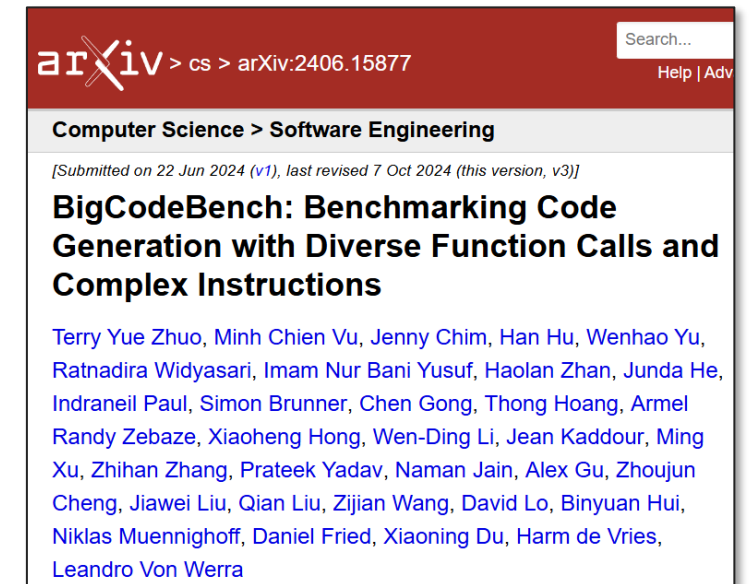
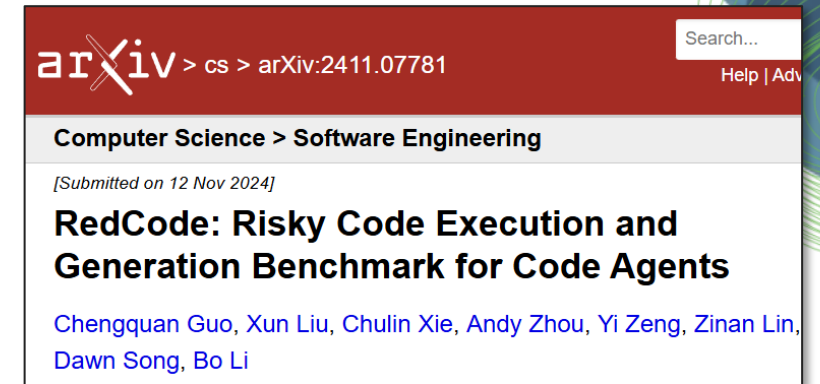
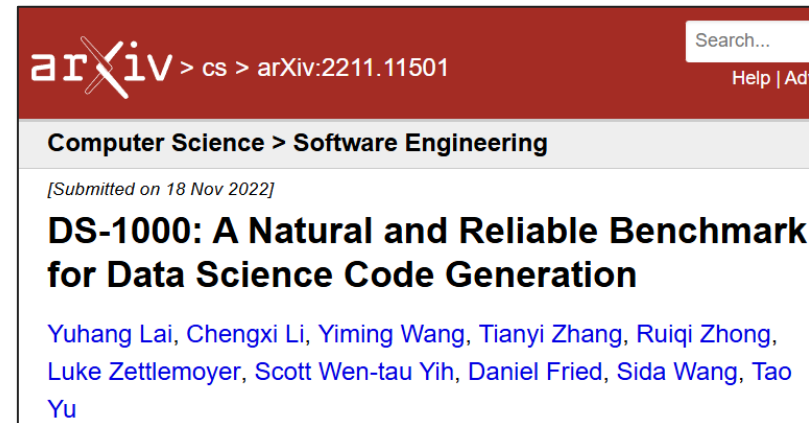
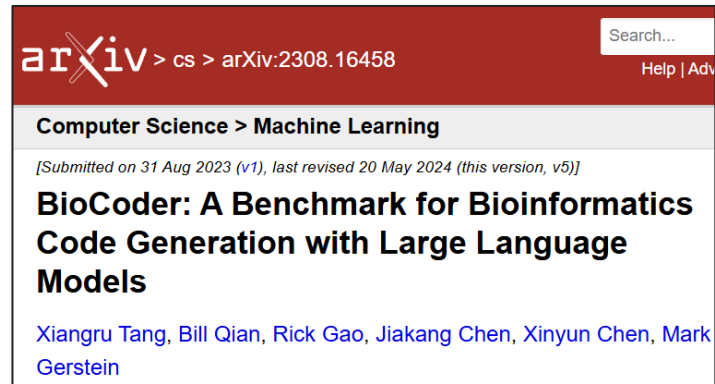
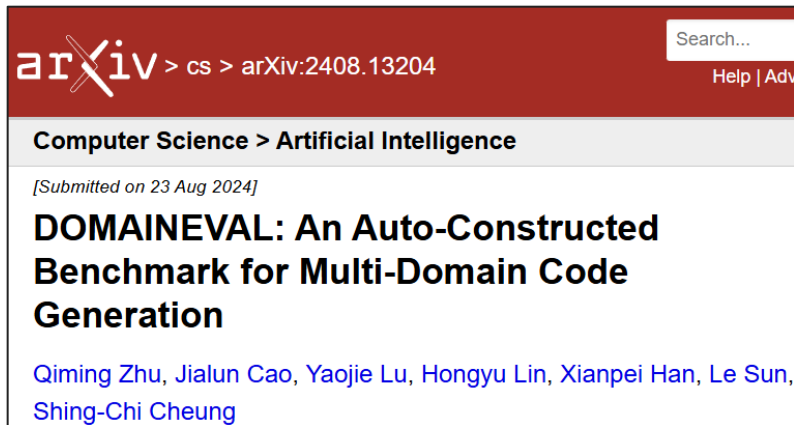
Multi-agent systems

- Multiple agents with
 - different specialities
 - Independent memory (inner chat history)
- Scheduler agent distributes tasks



Example: Summarizing Arxiv-Papers

- Task: Read 5 arxiv papers and write a review about them



Example: Summarizing Arxiv-Papers

- Task: Read 5 arxiv papers and write a review about them

Research assistant

Can:

- Read arxiv abstracts

Scheduler needs this information to decide about which agent can do which tasks

```
research_agent_factory = agent_factory(  
    name="research-assistant",  
    description="Scientific assistant who can read a paper and provide a summary of it.",  
    system_message="""You will be tasked to read paper(s) and provide a summary.  
Write a very detailed manuscript about 1000 words outlining the major messages and limitations of a given paper.""",  
    tools=[read_arxiv_paper],  
    model=model,  
    api_base=api_base,  
    api_key=api_key,  
)
```

```
@tool  
def read_arxiv_paper(arxiv_url:str)->str:  
    """Read the abstract of an arxiv-paper and return most important contents in markdown format.  
  
    Args:  
        arxiv_url: url of the Arxiv paper  
    """
```

The framework needs this information to set parameters accordingly

Example: Summarizing Arxiv-Papers

- Task: Read 5 arxiv papers and write a review about them

Reviewer

Can:

- Read text
- Provide feedback

```
@tool
def review_text(manuscript:str)->str:
    """Reviews text and provides constructive feedback

    Args:
        manuscript: complete original manuscript text to review.
    """

    feedback = prompt(f"""
You are a great reviewer and you like to provide constructive feedback.
If you are provided with a manuscript, you formulate feedback specifically for this manuscript.
Your goal is to guide the author towards writing a great manuscript.
Hence, provide feedback like these examples but focus on what makes sense for the given manuscript:
* a scientific text with a short and descriptive title,
* a scientific text with markdown sub-sections (# title, ## headlines, ...) avoiding bullet points,
* structured in sub-sections by content, e.g. introduction, recent developments, methods, results, discussion, future work, ...
* text using high-quality scientific language,
* proper citations mentioning the first author et al. using markdown links to original paper urls (do not make up references!),
* avoid mentioning "the paper" and use proper markdown-link-citations instead,
* a clear abstract at the beginning of the text, and conclusions at the end

## Manuscript
This is the manuscript you are asked to review:

{manuscript}

## Your task
Provide constructive feedback to the manuscript above.
""")

    return feedback
```

Example: Summarizing Arxiv-Papers

- Output

Evaluating Large Language Models in Code Generation: Recent Benchmarks and Future Directions

Abstract

The recent introduction of several new benchmarks has significantly advanced the evaluation of large language models (LLMs) in generating code. This manuscript provides an overview of these benchmarks, including [BigCodeBench](#), [DS-1000](#), [BioCoder](#), [RedCode](#), and [DOMAINEVAL](#), and discusses their implications for the development of LLMs.

Introduction

The recent papers introduce several new benchmarks for evaluating the performance of large language models (LLMs) in generating code. These benchmarks include [BigCodeBench](#), [DS-1000](#), [BioCoder](#), [RedCode](#), and [DOMAINEVAL](#). The results of these papers show that LLMs are not yet capable of following complex instructions to use function calls precisely and struggle with certain tasks such as cryptography and system coding. However, they also demonstrate the potential of LLMs in generating bioinformatics-specific code and highlight the importance of domain-specific knowledge.

Discussion

Overall, these benchmarks provide a challenging and reliable evaluation platform for data science code generation models and emphasize the need for further research and development. The results of these benchmarks have significant implications for the development of LLMs, highlighting the need for improved performance in following complex instructions and generating code for specific domains. Furthermore, the benchmarks demonstrate the potential of LLMs in generating high-quality code for certain tasks, such as bioinformatics, and emphasize the importance of incorporating domain-specific knowledge into LLMs.

Conclusion

In conclusion, the recent benchmarks for evaluating LLMs in code generation have provided significant insights into the capabilities and limitations of these models. The results of these benchmarks highlight the need for further research and development to improve the performance of LLMs in generating code, particularly in areas such as cryptography and system coding. However, they also demonstrate the potential of LLMs in generating high-quality code for specific domains, such as bioinformatics, and emphasize the importance of incorporating domain-specific knowledge into these models.

Model Context Protocol (MCP)

- Quite successful approach to standardize communication between agents and backends

Client

Chat interface
Claude Desktop, LibreChat

MCP
Standardized protocol

Bidirectional
data flow

AI applications

Server

Data and file systems
PostgreSQL, SQLite, GDrive

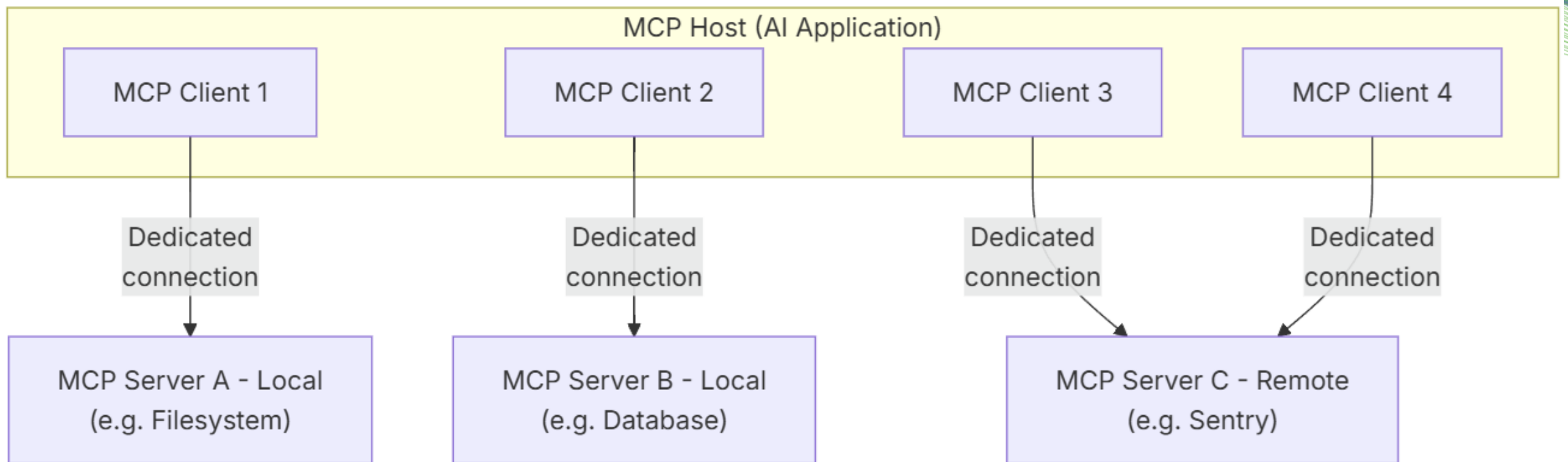
Development tools
Git, Sentry, etc.

Productivity tools
Slack, Google Maps, etc.

Bidirectional
data flow

Data sources and tools

Model Context Protocol (MCP)



MCP Server

- Expose your data / tools to the world

```
from mcp.server.fastmcp import FastMCP
mcp = FastMCP("mensa-tud-mcp")
```

```
@mcp.tool()
```

```
async def list_canteens() -> str:
```

```
    """List all available canteens from the Studentenwerk Dresden.
```

```
    Returns canteen IDs, names, addresses, and coordinates for all available canteens.
```

```
    """
```


MCP Server

- Expose your data / tools to the world

```
@mcp.tool()
```

```
async def get_meals(canteen_id: int, date: str | None = None) -> str:  
    """Get all meals available at a specific canteen on a specific date.
```

Notes:

- * To figure out the ID of a specific canteen, first use the `list_canteens` tool.
- * Also make sure to check which days have meal data available using the `list_canteen_days` tool.

Args:

canteen_id: The ID of the canteen to query (e.g., 9 for Alte Mensa)

date: The date in YYYY-MM-DD format (e.g., 2026-01-13). If not provided, uses today's date.

Returns meal names, categories, prices, and dietary information.

```
"""
```

Guide the LLM in
which order to
call functions

MCP Server

- Endless MCP Servers published recently

The screenshot shows a YouTube video player. The video title is "What is the Figma MCP server?". The main content is a Figma canvas displaying a "Game flow chart" with various nodes and connectors. A chat window is overlaid on the right side of the video, showing a conversation with an AI agent. The chat text includes: "I'd like to create a game from this FigJam. My selection contains the logic for the game. Please implement a simple web app where you can feed or pet the figural (you can just use buttons for now). Don't worry about the assets for now, I'll point you there later." and a URL: "https://www.figma.com/board/1BvVGWwVYmUkZ3n5CjS/Untitled?node-id=1-499&t=6fLeM3Bnf8WvTw3y-4". The video player controls at the bottom show a duration of 2:00 / 5:17 and the title "Figma MCP and FigJam".

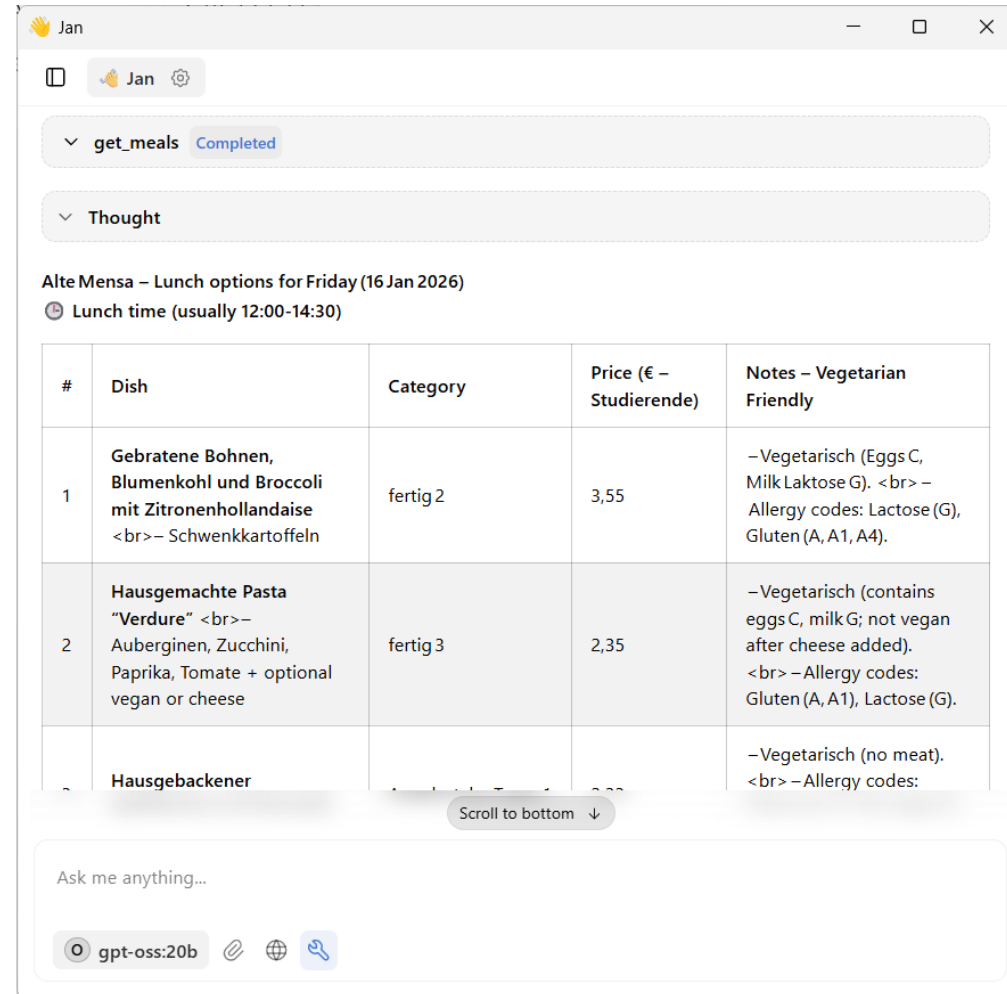
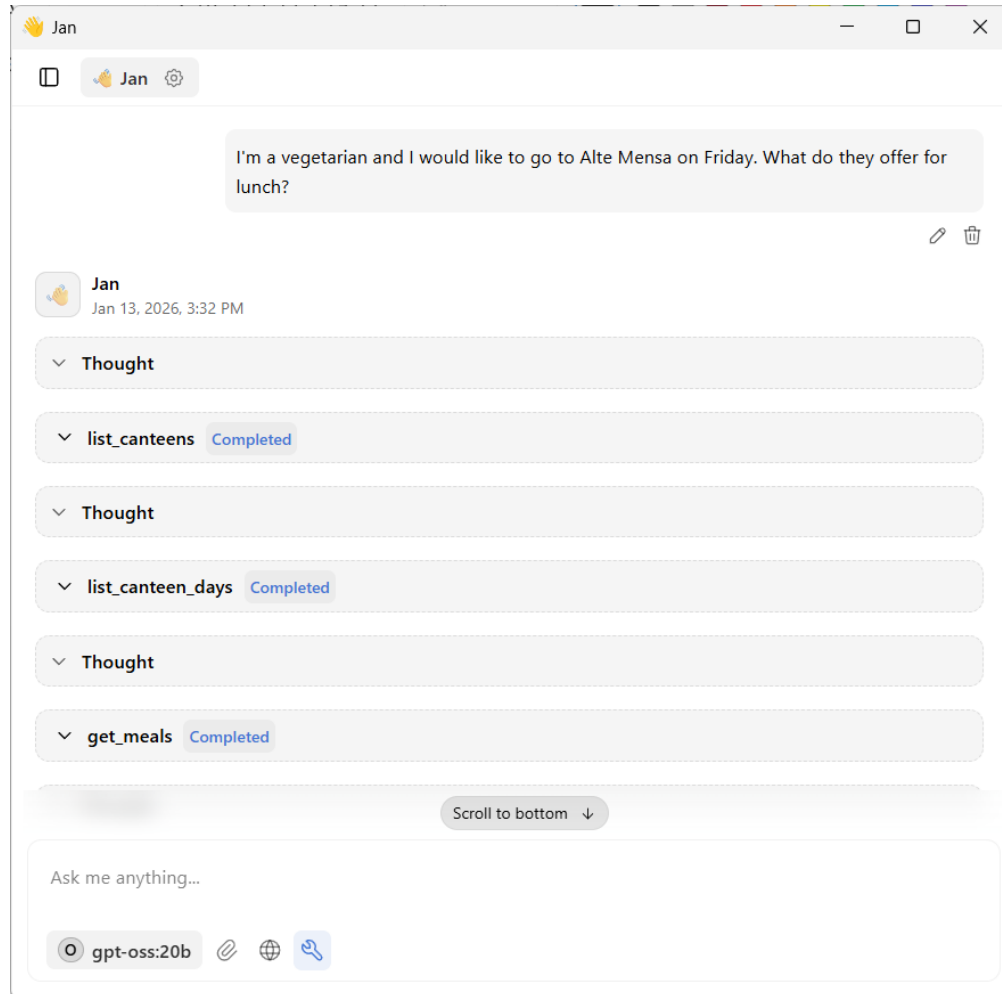
The screenshot shows the Microsoft Learn website. The browser address bar displays the URL: "learn.microsoft.com/en-us/microsoft-agent-365/mcp-server-reference/calendar". The page title is "Microsoft Outlook Calendar MCP server reference". The page content includes a table of contents with links to "Overview", "Discover and use", "Manage", "Observe", "Security and compliance", "Integrate with", and "Agent 365 development". The "Overview" section is currently selected. The page also features a search bar, a "Sign in" button, and a "Learn / Microsoft Agent 365 /" breadcrumb trail.

MCP Server – Client connection

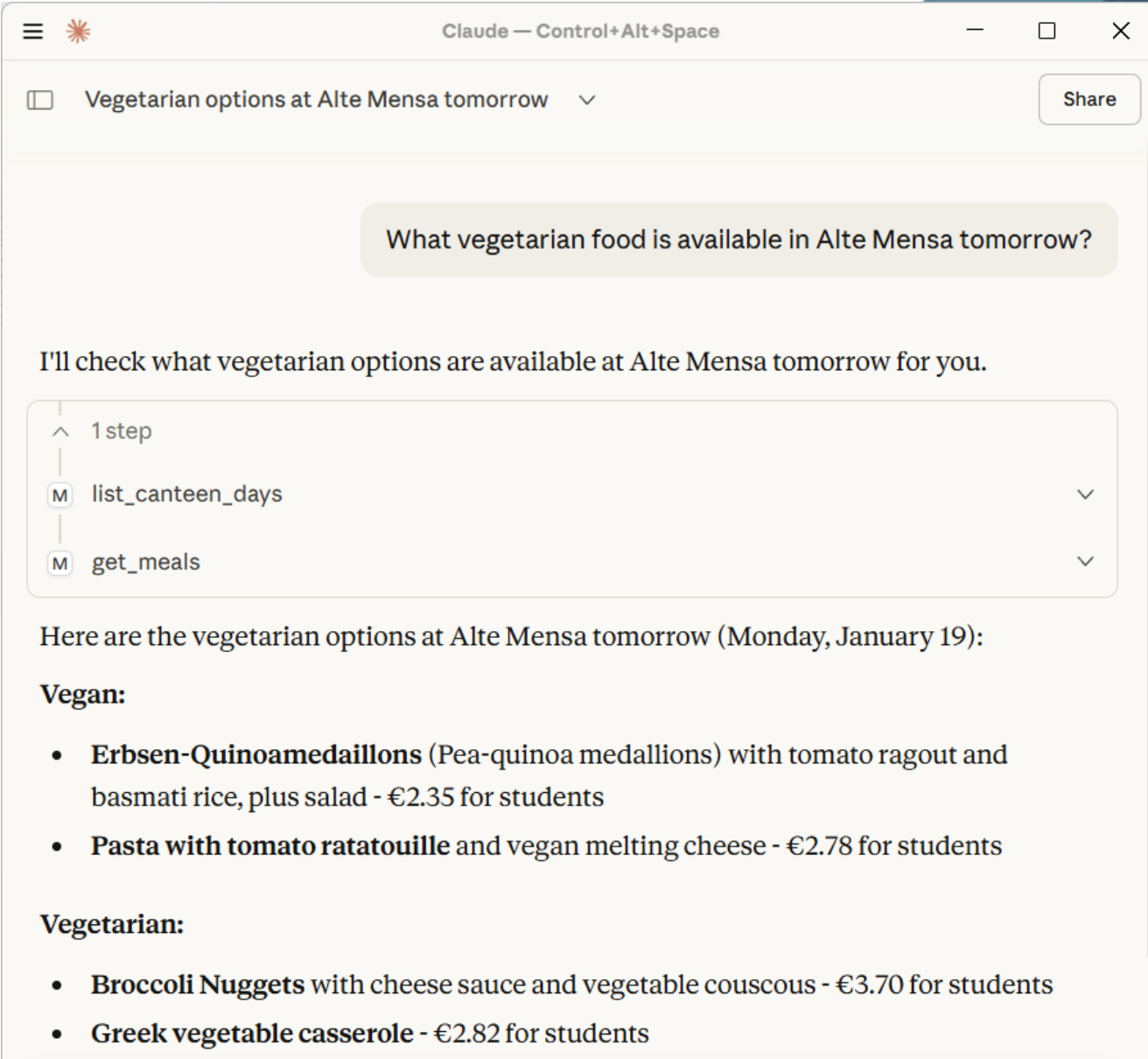
- Configure your server within a client:

```
"mensa-tud": {  
  "args": [  
    "--directory",  
    "c:\\structure\\code\\mensa-tud-mcp",  
    "run",  
    "server.py"  
  ],  
  "command": "uv"  
},
```

MCP Clients: Jan AI (open source)



MCP Clients: Claude (commercial)



The screenshot shows the Claude web interface. At the top, the title bar says "Claude — Control+Alt+Space". Below it, a search bar contains the text "Vegetarian options at Alte Mensa tomorrow" with a dropdown arrow and a "Share" button. The main chat area shows a user query: "What vegetarian food is available in Alte Mensa tomorrow?". The assistant's response is: "I'll check what vegetarian options are available at Alte Mensa tomorrow for you." Below this, a "1 step" section shows two tool calls: "list_canteen_days" and "get_meals", each with a small "M" icon and a dropdown arrow. The assistant then provides the results: "Here are the vegetarian options at Alte Mensa tomorrow (Monday, January 19):". Under the heading "Vegan:", there are two bullet points: "Erbsen-Quinoamedaillons (Pea-quinoa medallions) with tomato ragout and basmati rice, plus salad - €2.35 for students" and "Pasta with tomato ratatouille and vegan melting cheese - €2.78 for students". Under the heading "Vegetarian:", there are two bullet points: "Broccoli Nuggets with cheese sauce and vegetable couscous - €3.70 for students" and "Greek vegetable casserole - €2.82 for students".

Claude — Control+Alt+Space

Vegetarian options at Alte Mensa tomorrow

Share

What vegetarian food is available in Alte Mensa tomorrow?

I'll check what vegetarian options are available at Alte Mensa tomorrow for you.

1 step

- list_canteen_days
- get_meals

Here are the vegetarian options at Alte Mensa tomorrow (Monday, January 19):

Vegan:

- **Erbsen-Quinoamedaillons** (Pea-quinoa medallions) with tomato ragout and basmati rice, plus salad - €2.35 for students
- **Pasta with tomato ratatouille** and vegan melting cheese - €2.78 for students

Vegetarian:

- **Broccoli Nuggets** with cheese sauce and vegetable couscous - €3.70 for students
- **Greek vegetable casserole** - €2.82 for students

Limitations of agentic workflows

- If agents had a success-rate of 99%, and we chain 5 agents in a workflow, our workflow's success rate: $0.99^5 = 95\%$
 - **Commercial models** may actually have a success-rate of 99% when calling a function.
 - **Open weight models may not.** Current reasonable options: Jan-VL, gptoss
- If agents had a success-rate of 90%, and we chain 5 agents in a workflow, our workflow's success rate: $0.9^5 = 59\%$
- If agents had a success-rate of 50%, and we chain 5 agents in a workflow, our workflow's success rate: $0.5^5 = 3\%$
- Many frameworks solve this problem by calling functions in loops and checking results, e.g. for consistency -> **high costs!**

Summary

- LLMs can generate code and decide for functions to call depending on a task -> non-deterministic program flow
 - flexibility
 - reproducibility
- Agentic systems
 - Use function / tool calling to enrich the context with information
 - Could execute actions (potentially causing costs, cause trouble)
- Limits of agents
 - Ethical use
 - Work best with commercial LLMs (-> privacy?)
 - Success probability