CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Large Language Models for Function Calling

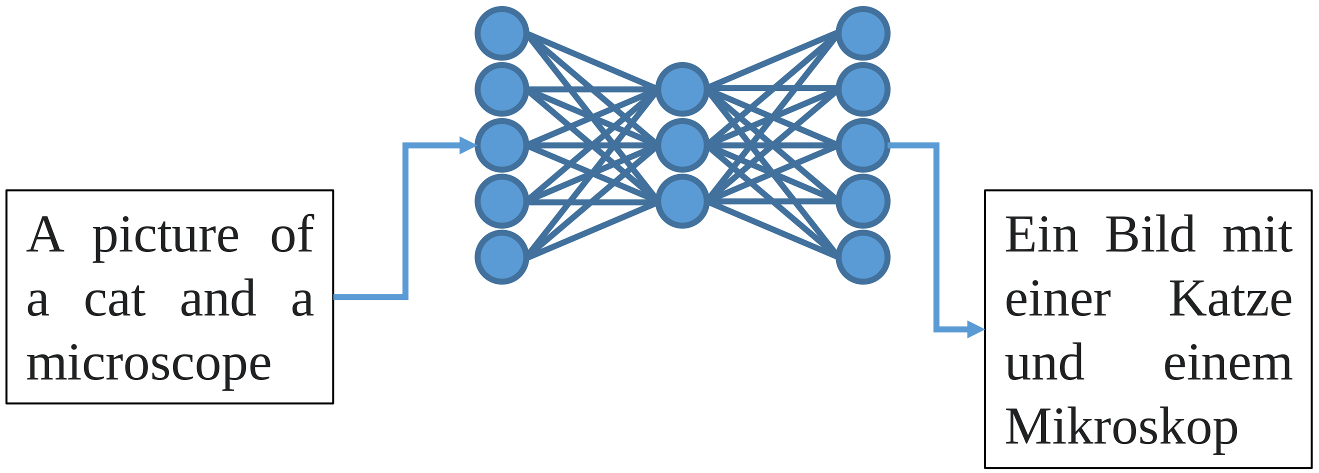
Robert Haase

Generative Artificial Intelligence

- Definition: “Generative artificial intelligence […] is a type of artificial intelligence (AI) system capable of generating text, images, or other media in response to prompts.”[1]

- Commonly based on Neural Networks
- Bridges fields:
  - Natural Language Processing (NLP)
  - Computer Vision (CV)

- Use-cases
  - Translating text
  - Writing emails, text, grant proposals
  - Summarizing articles
  - Writing code
  - General question answering
  - Image generation
  - Image interpretation / analysis


A picture of a cat and a microscope
Ein Bild mit einer Katze und einem Mikroskop

Robert Haase
@haesleinhuepf
BIDS Lecture 10/14
June 4th 2024

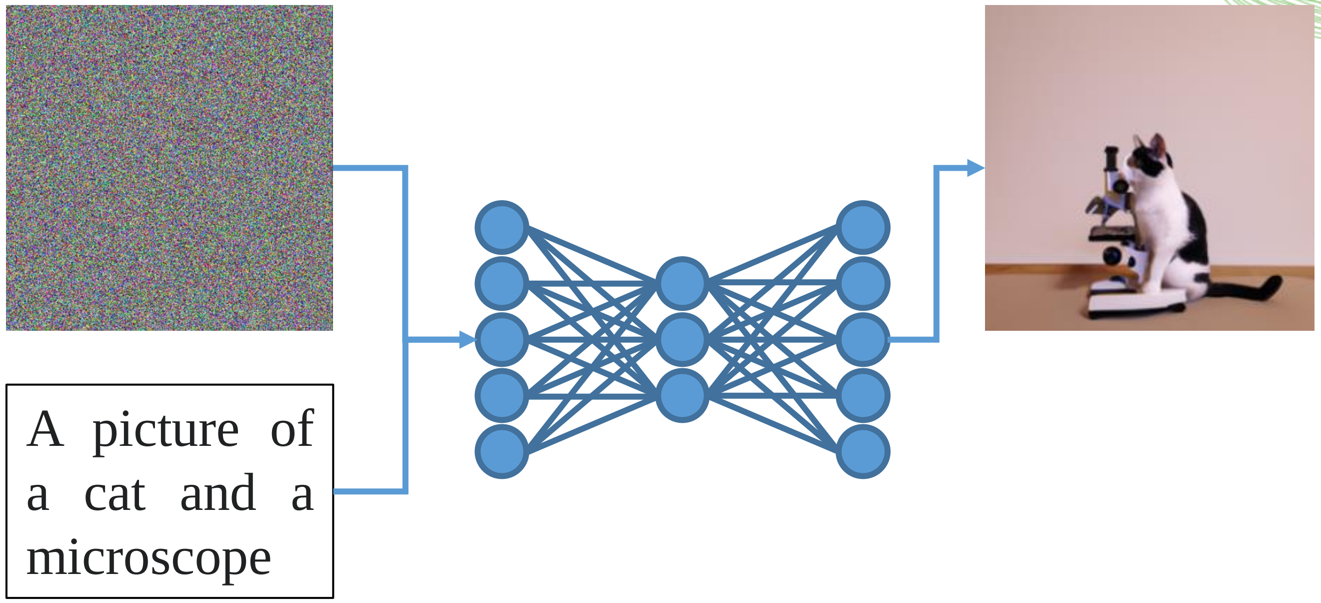
[1] source: https://en.wikipedia.org/wiki/Generative_artificial_intelligence

Generative Artificial Intelligence

- Definition: "Generative artificial intelligence […] is a type of artificial intelligence (AI) system capable of generating text, images, or other media in response to prompts."[1]

- Commonly based on Neural Networks
- Bridges fields:
  - Natural Language Processing (NLP)
  - Computer Vision (CV)

- Use-cases
  - Translating text
  - Writing emails, text, grant proposals
  - Summarizing articles
  - Writing code
  - General question answering
  - Image generation
  - Image interpretation / analysis



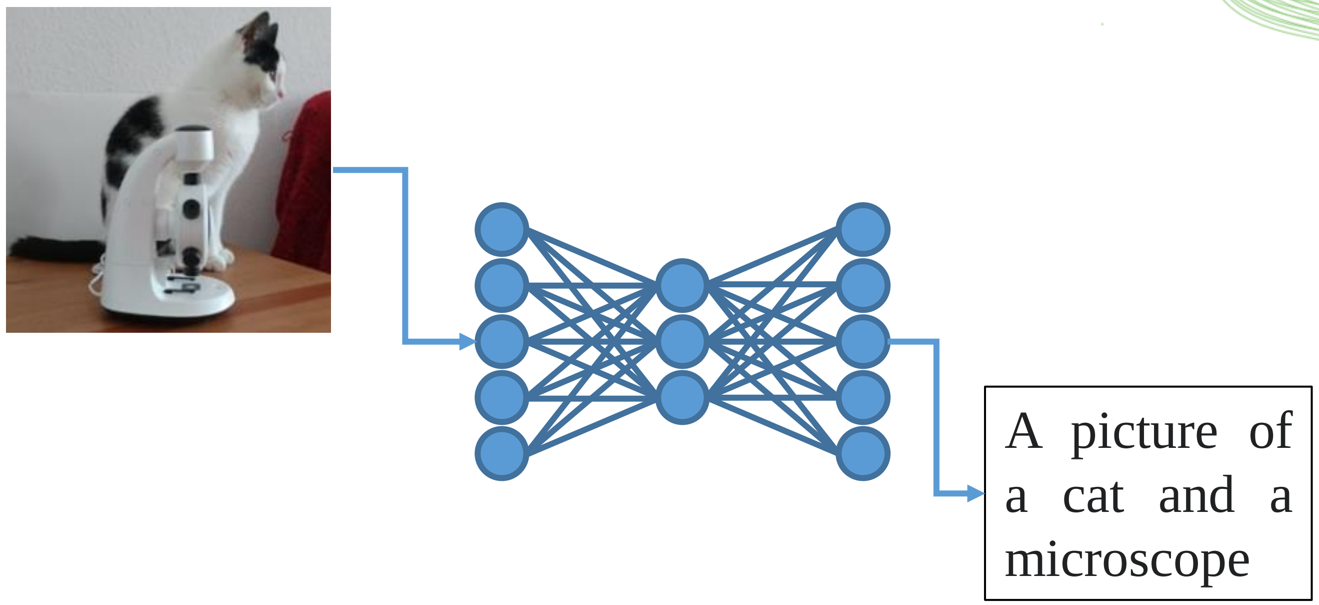
[1] source: https://en.wikipedia.org/wiki/Generative_artificial_intelligence

Generative Artificial Intelligence

- Definition: “Generative artificial intelligence […] is a type of artificial intelligence (AI) system capable of generating text, images, or other media in response to prompts.”[1]

- Commonly based on Neural Networks
- Bridges fields:
  - Natural Language Processing (NLP)
  - Computer Vision (CV)

- Use-cases
  - Translating text
  - Writing emails, text, grant proposals
  - Summarizing articles
  - Writing code
  - General question answering
  - Image generation
  - Image interpretation / analysis



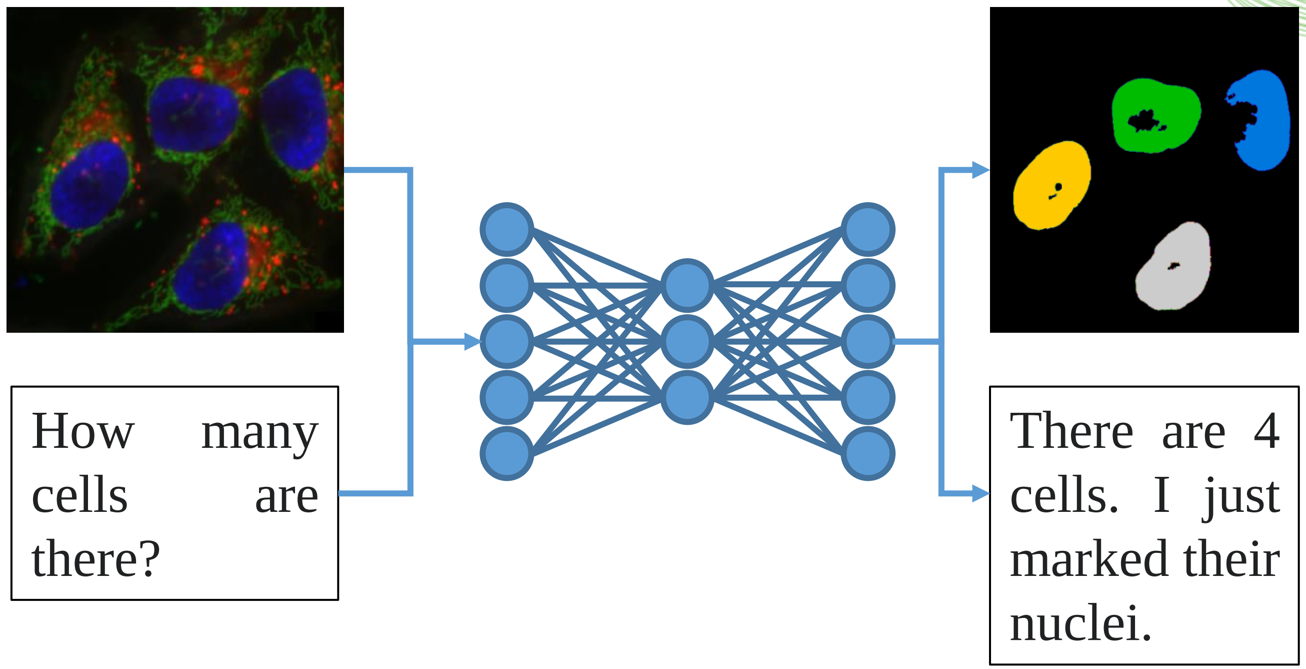
[1] source: https://en.wikipedia.org/wiki/Generative_artificial_intelligence

Generative Artificial Intelligence

- Definition: "Generative artificial intelligence [...] is a type of artificial intelligence (AI) system capable of generating text, images, or other media in response to prompts."[1]

- Commonly based on Neural Networks
- Bridges fields:
  - Natural Language Processing (NLP)
  - Computer Vision (CV)

- Use-cases
  - Translating text
  - Writing emails, text, grant proposals
  - Summarizing articles
  - Writing code
  - General question answering
  - Image generation
  - Image interpretation / analysis

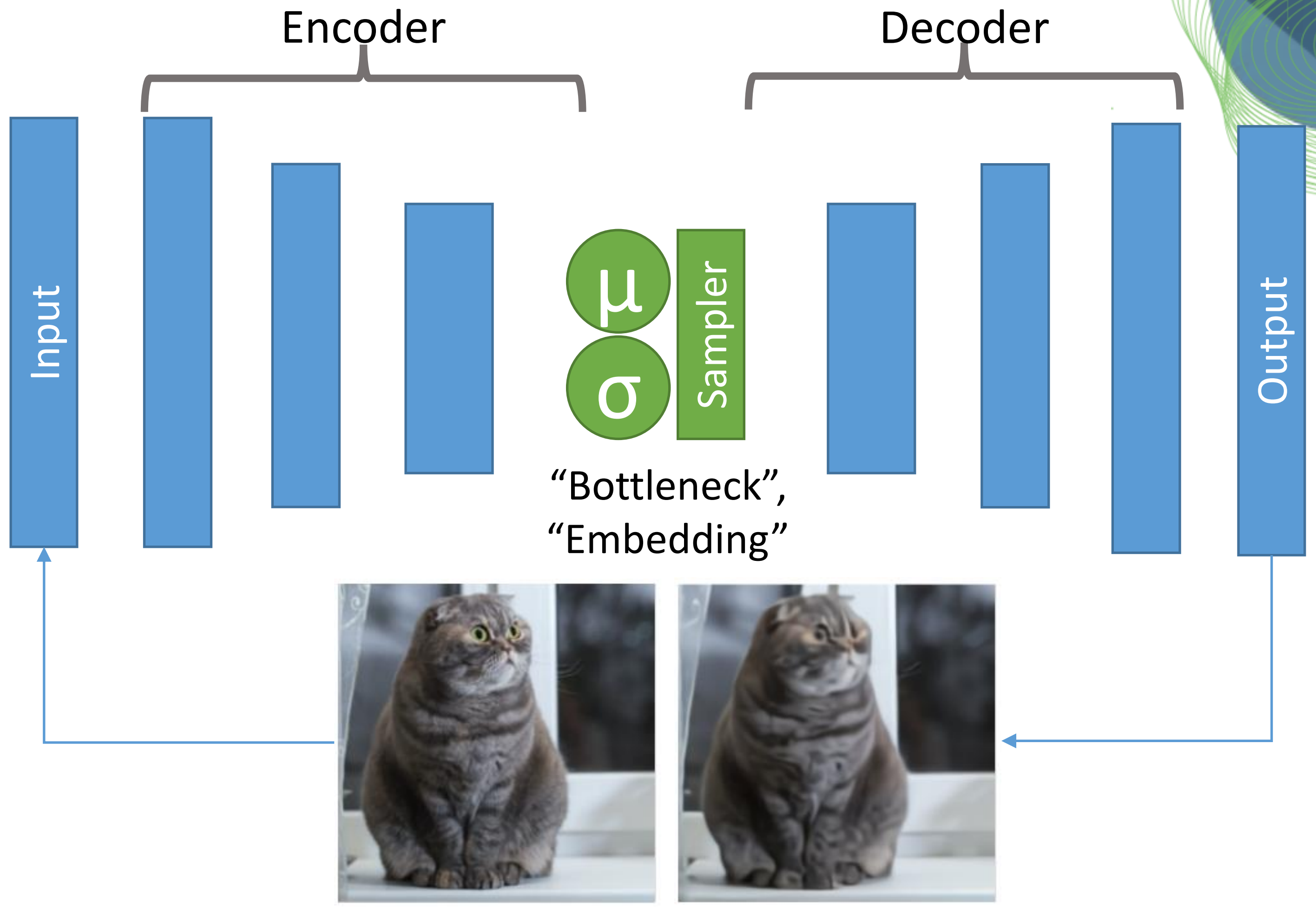

How many cells are there?
There are 4 cells. I just marked their nuclei.

Robert Haase
@haesleinhuepf
BIDS Lecture 10/14
June 4th 2024

[1] source: https://en.wikipedia.org/wiki/Generative_artificial_intelligence

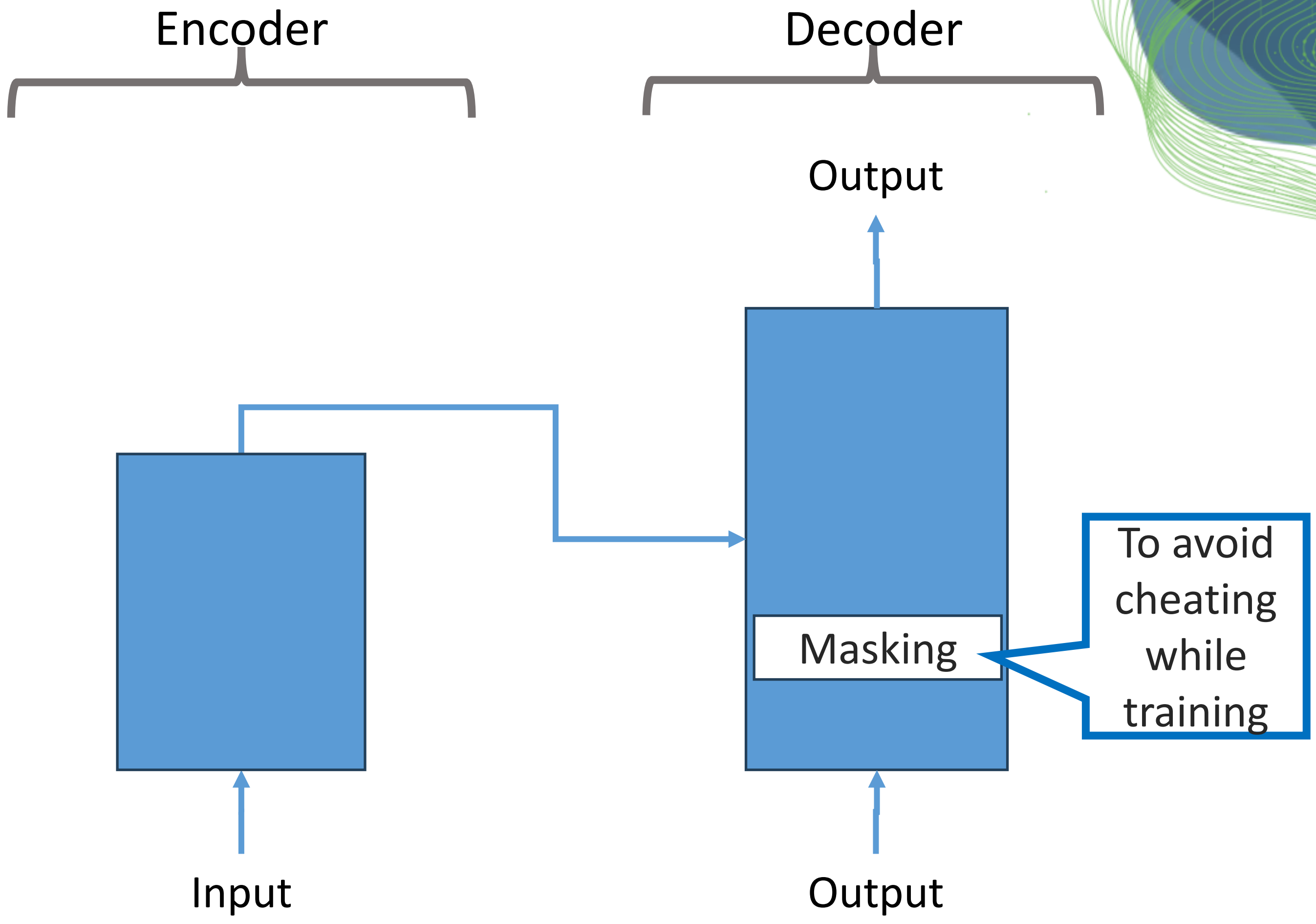
Variational Auto-Encoder

- Turning pixels into „meaning“ and back to pixels.



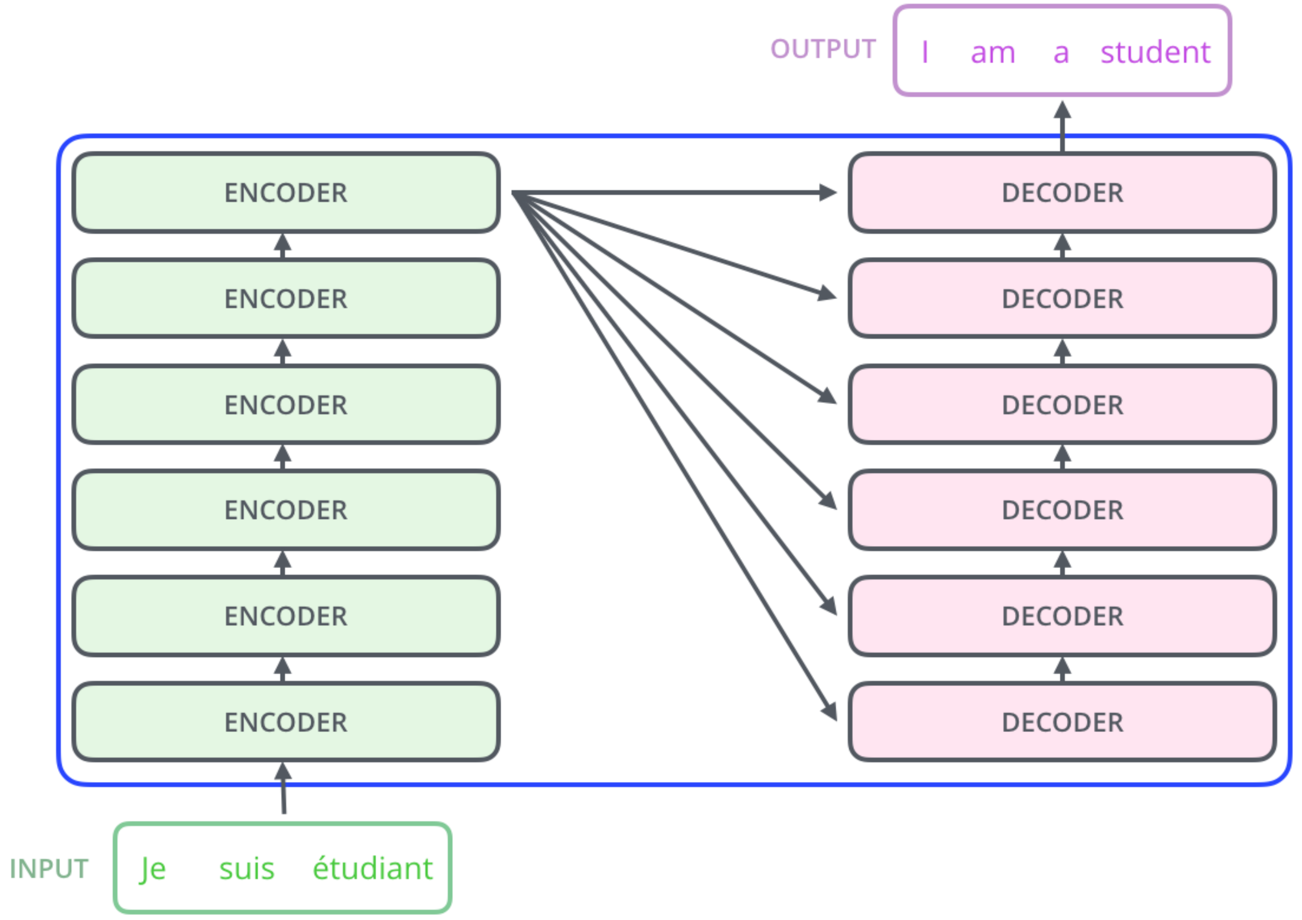
Cat picture source: Ramesh et al., licensed CC-BY 4.0
https://arxiv.org/pdf/2102.12092.pdf

Generative Pretrained Transformer (GPT)



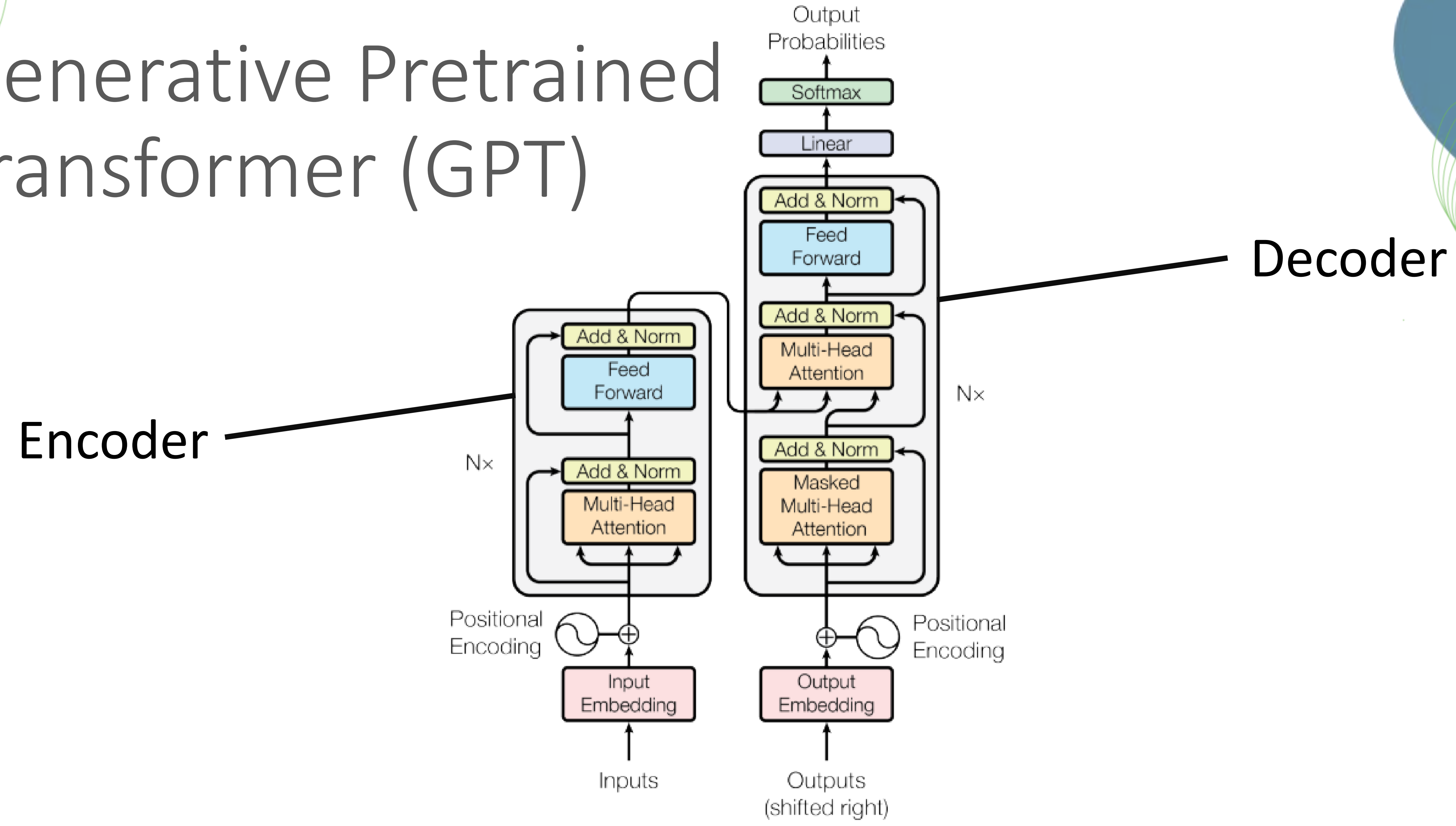
Generative Pretrained Transformer (GPT)

- Stacks of encoders and decoders arranged like this:



Cropped from Alammar (2018), licensed [CC-BY-SA 4.0](). [https://jalammar.github.io/illustrated-transformer/]()

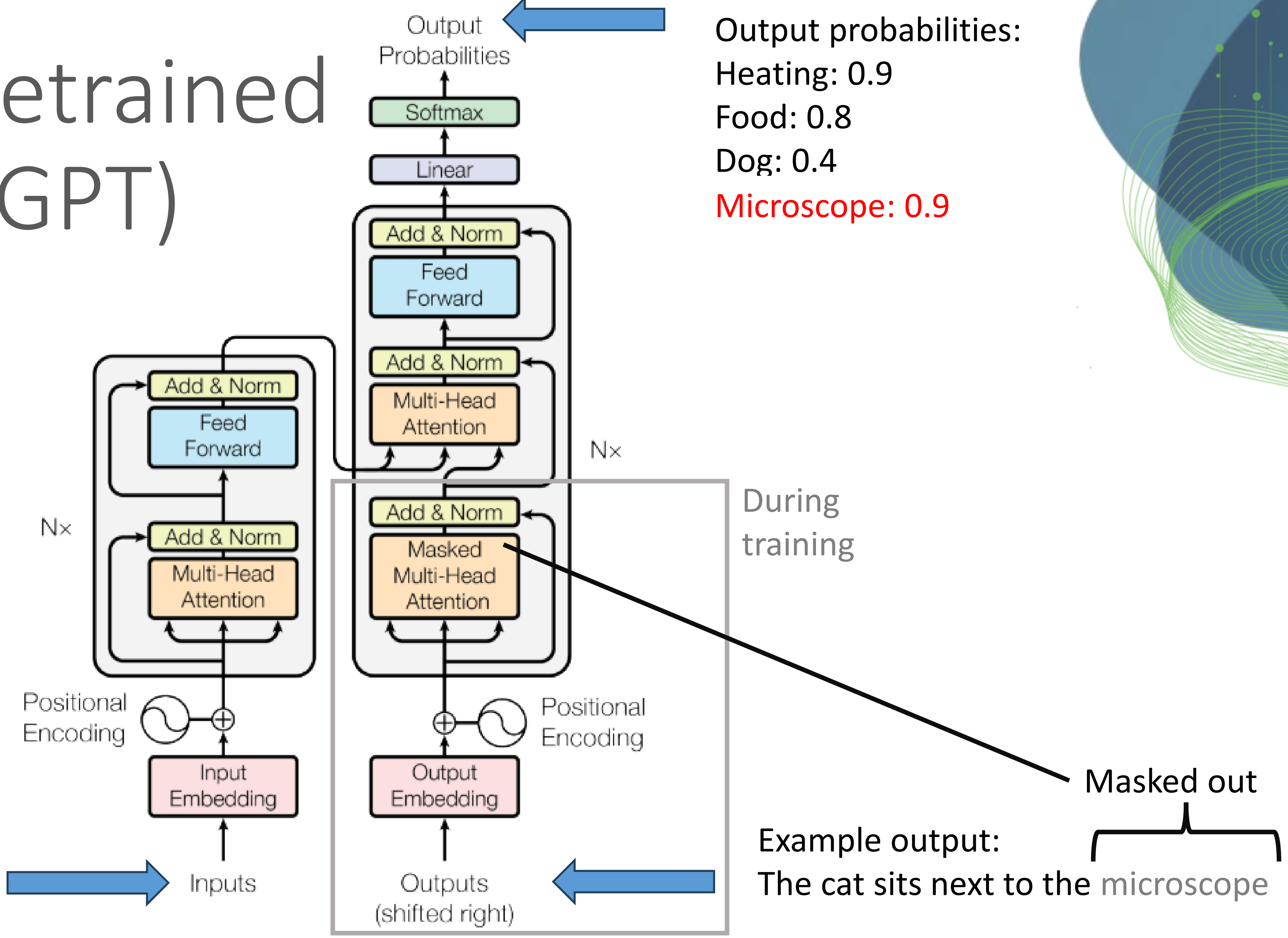
Generative Pretrained Transformer (GPT)



Source: Vaswani et al (2017)
https://arxiv.org/abs/1706.03762

Generative Pretrained Transformer (GPT)

- Task: Translation



Output probabilities:
Heating: 0.9
Food: 0.8
Dog: 0.4
Microscope: 0.9

Example input:
Die Katze sitzt neben dem Mikroskop.

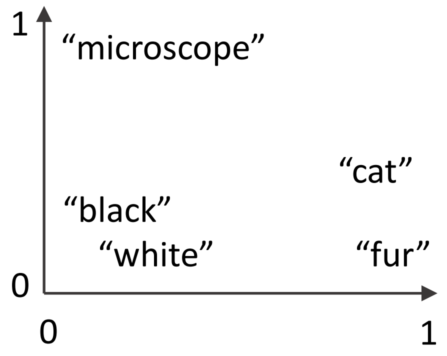
Example output:
The cat sits next to the microscope

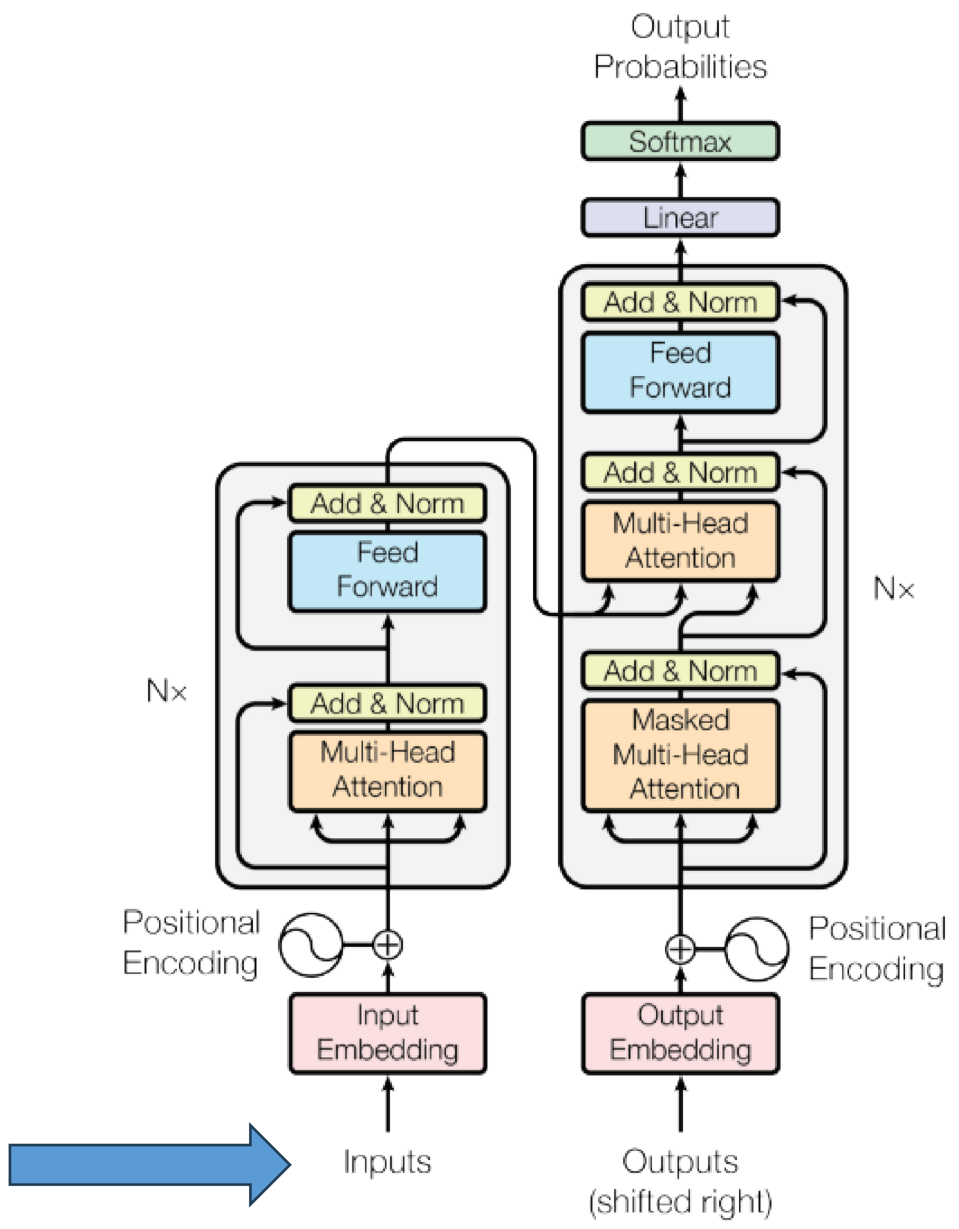
Source: Vaswani et al (2017)
https://arxiv.org/abs/1706.03762

Generative Pretrained Transformer (GPT)

- Words need to be converted into vectors to enable NNs to process them.

Word Embedding





Source: Vaswani et al (2017) https://arxiv.org/abs/1706.03762

Attention is all you need

- The position of the word in the sentence / context may have influence on its meaning.

The cat sits next to a microscope.

Next to the microscope there is a cat.

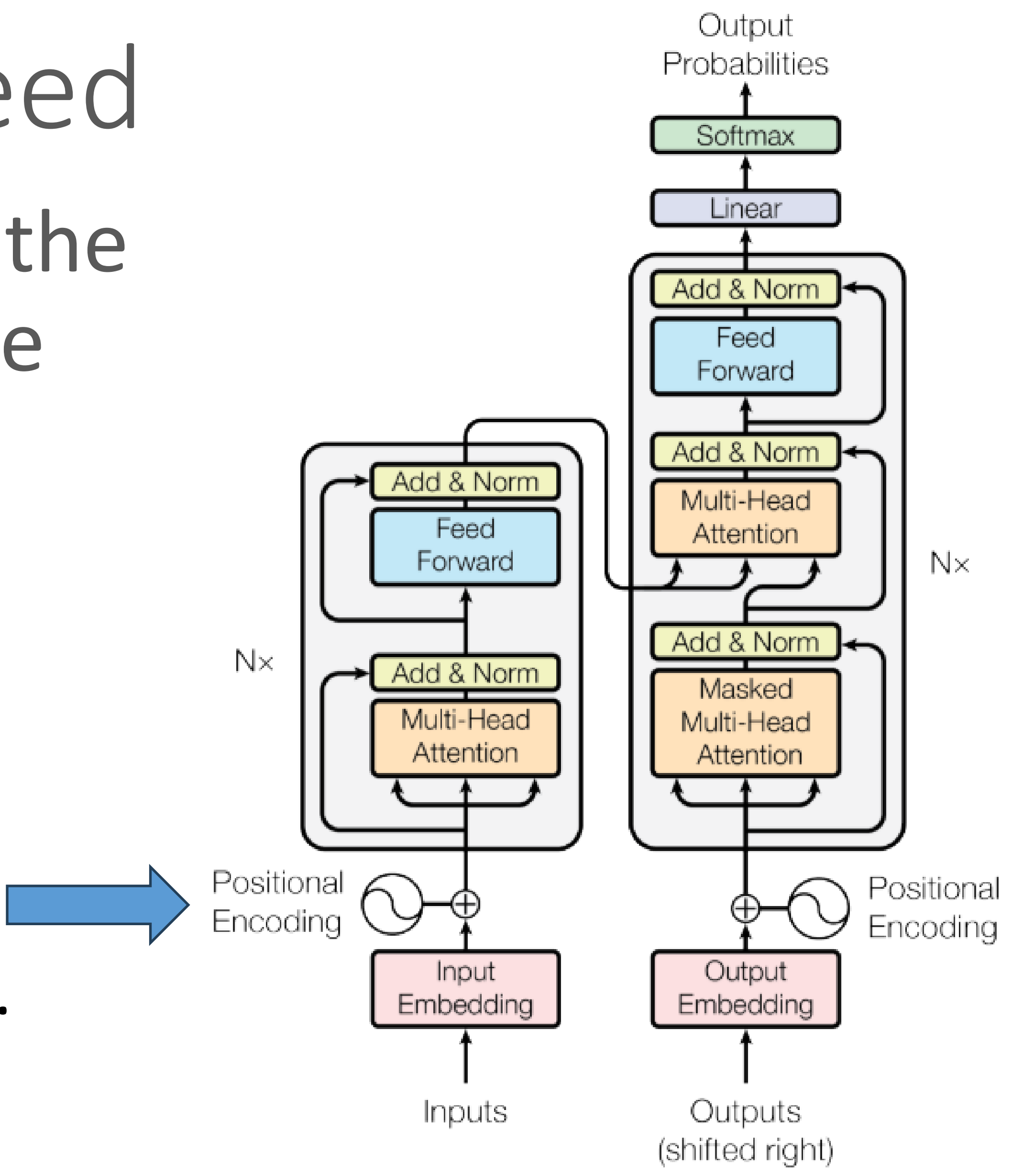
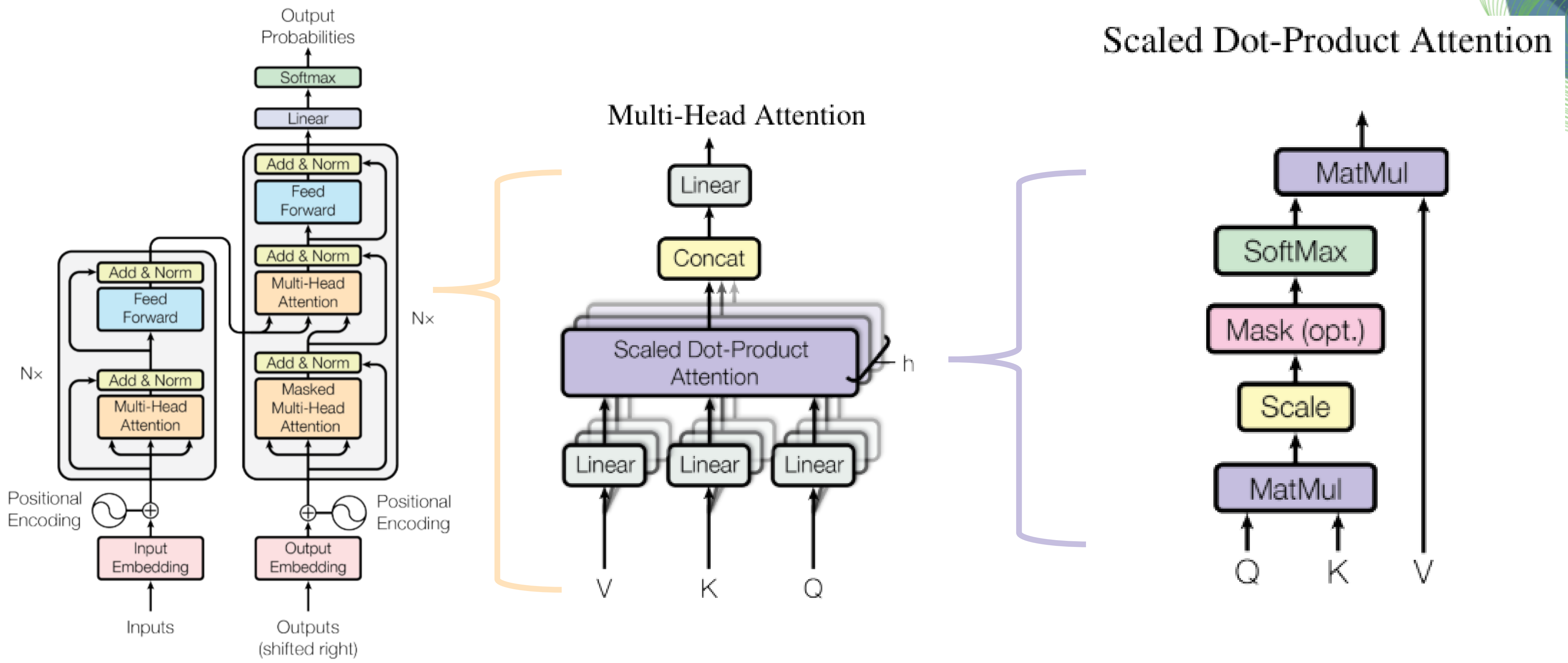


Figure source: Vaswani et al (2017)
https://arxiv.org/abs/1706.03762

Attention is all you need



Source: Vaswani et al (2017)
https://arxiv.org/abs/1706.03762

Scaled dot-product attention

- Attention score: How much related are two words?
- Query: For which word are we calculating attention?
- Key: To which word are we calculating attention
- Value: Relevance of the query-key relationship

The cat is black and white.

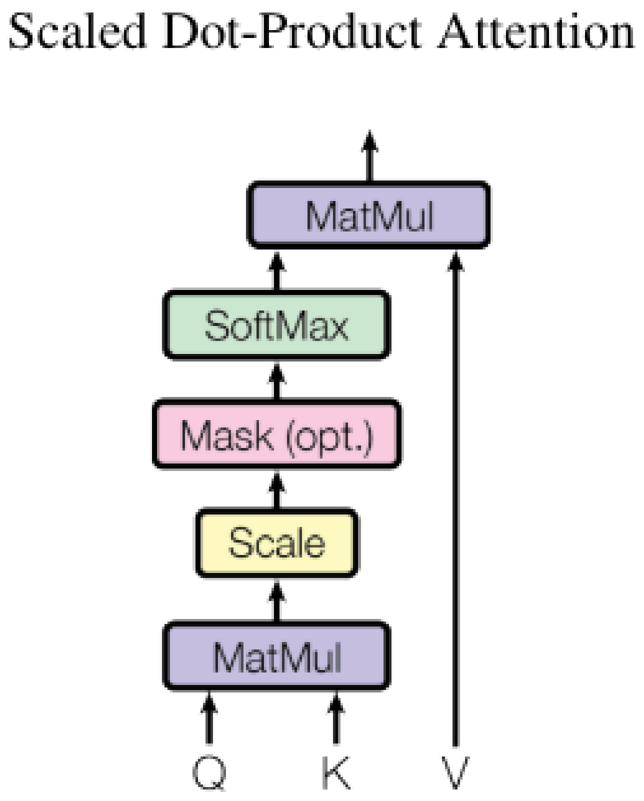
attention score

Relevance value: 0.1

The cat is meowing.

attention score

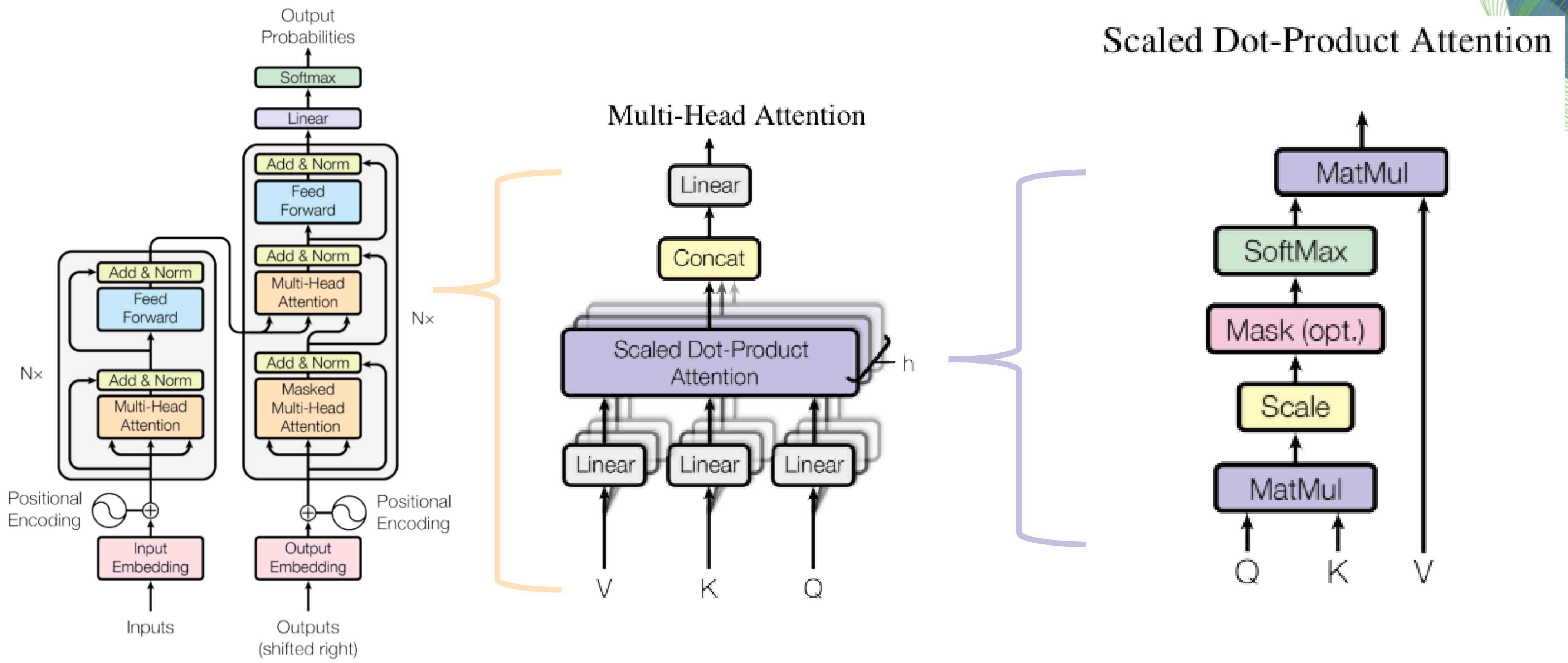
Relevance value: 0.9



Robert Haase
@haesleinhuepf
BIDS Lecture 10/14
June 4th 2024

Source: Vaswani et al (2017)
https://arxiv.org/abs/1706.03762
See also: https://www.youtube.com/watch?v=sznZ78HquPc

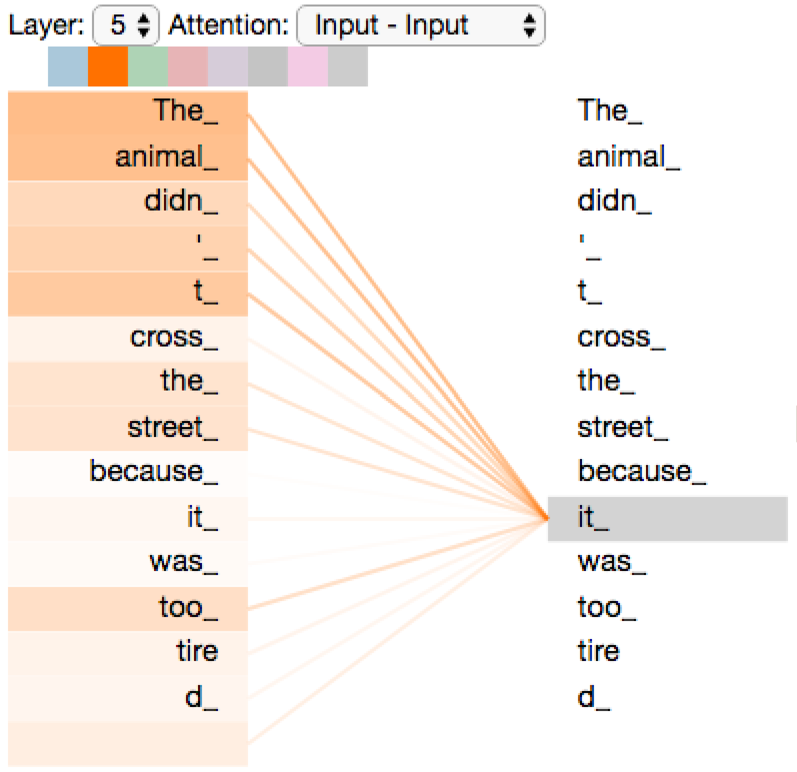
Attention is all you need

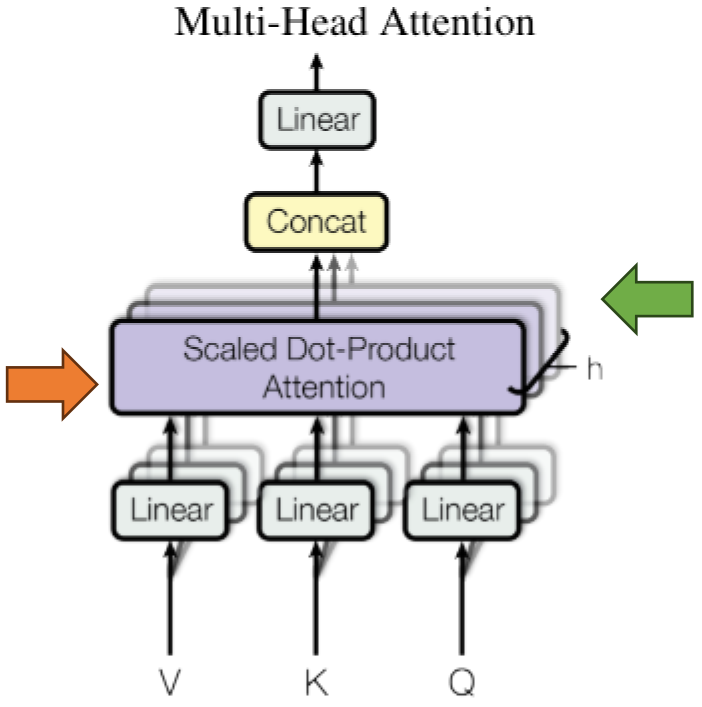


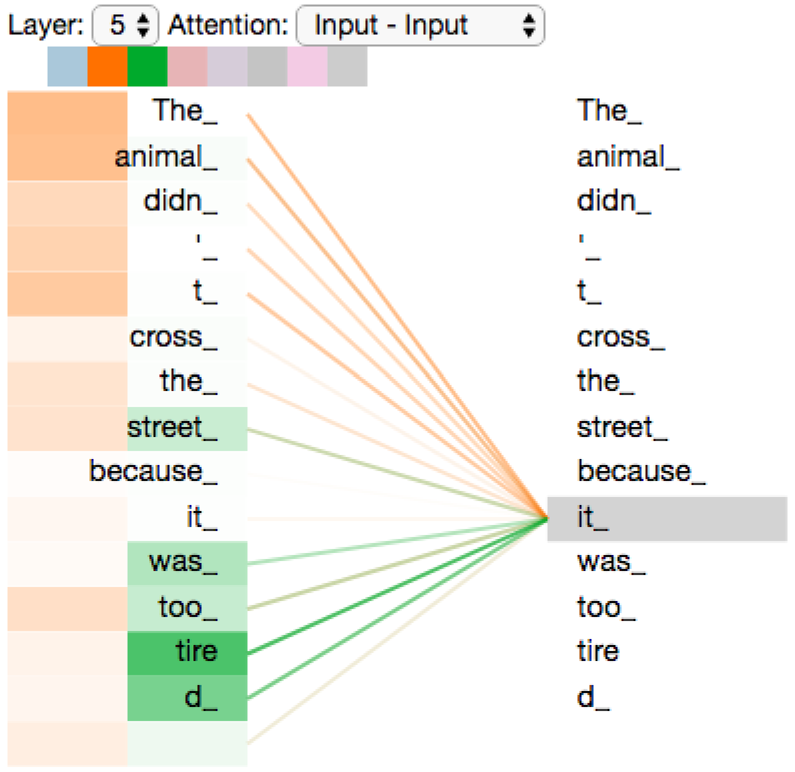
Source: Vaswani et al (2017)
https://arxiv.org/abs/1706.03762

Multi-head attentions

- Multiple aspects represented by multiple attention heads

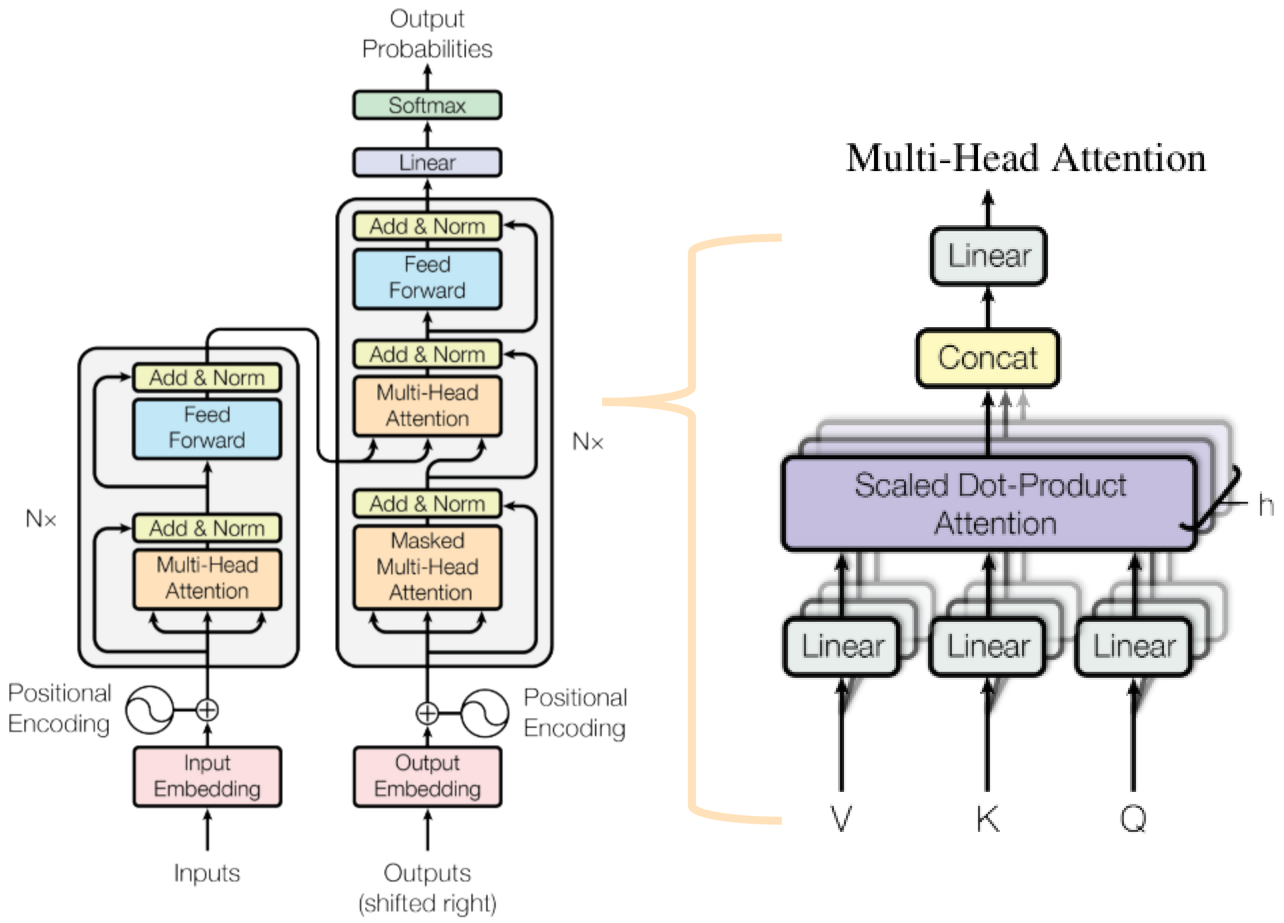


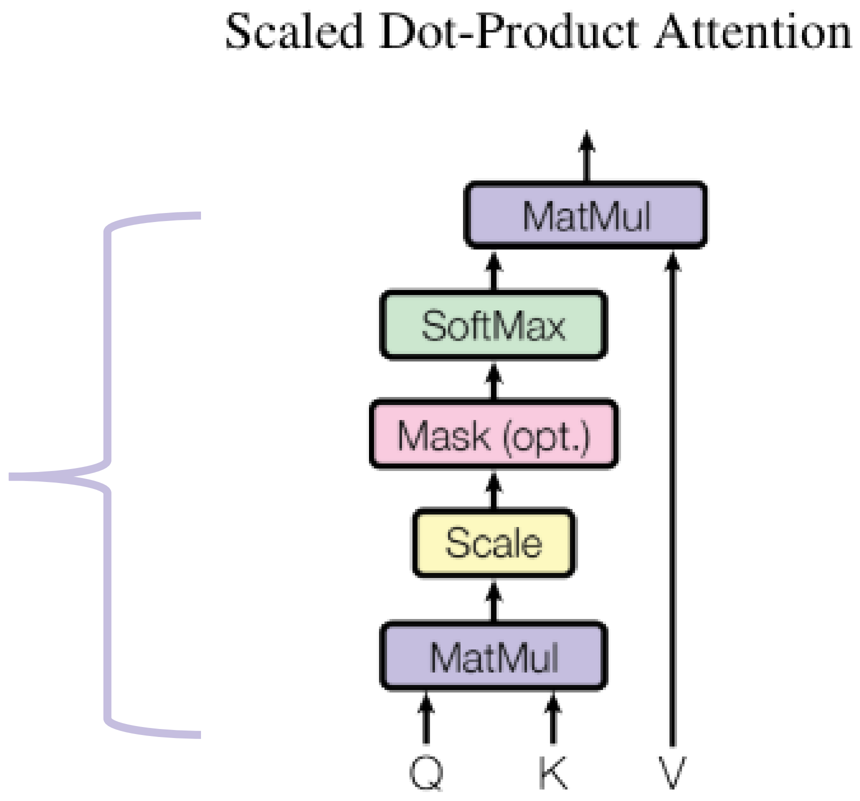




Source Figure left/right: Alammar (2018), licensed CC-BY-SA 4.0.
https://jalammar.github.io/illustrated-transformer/
Source Figure center: Vaswani et al (2017)
https://arxiv.org/abs/1706.03762

Attention is all you need

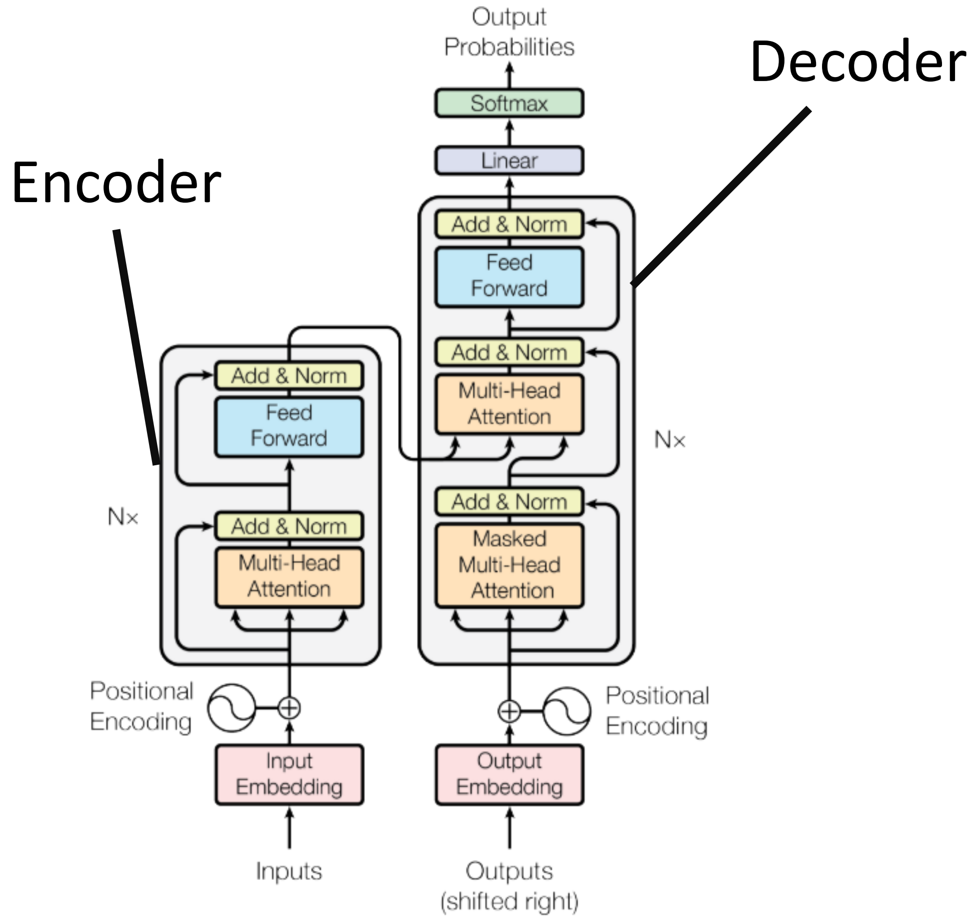


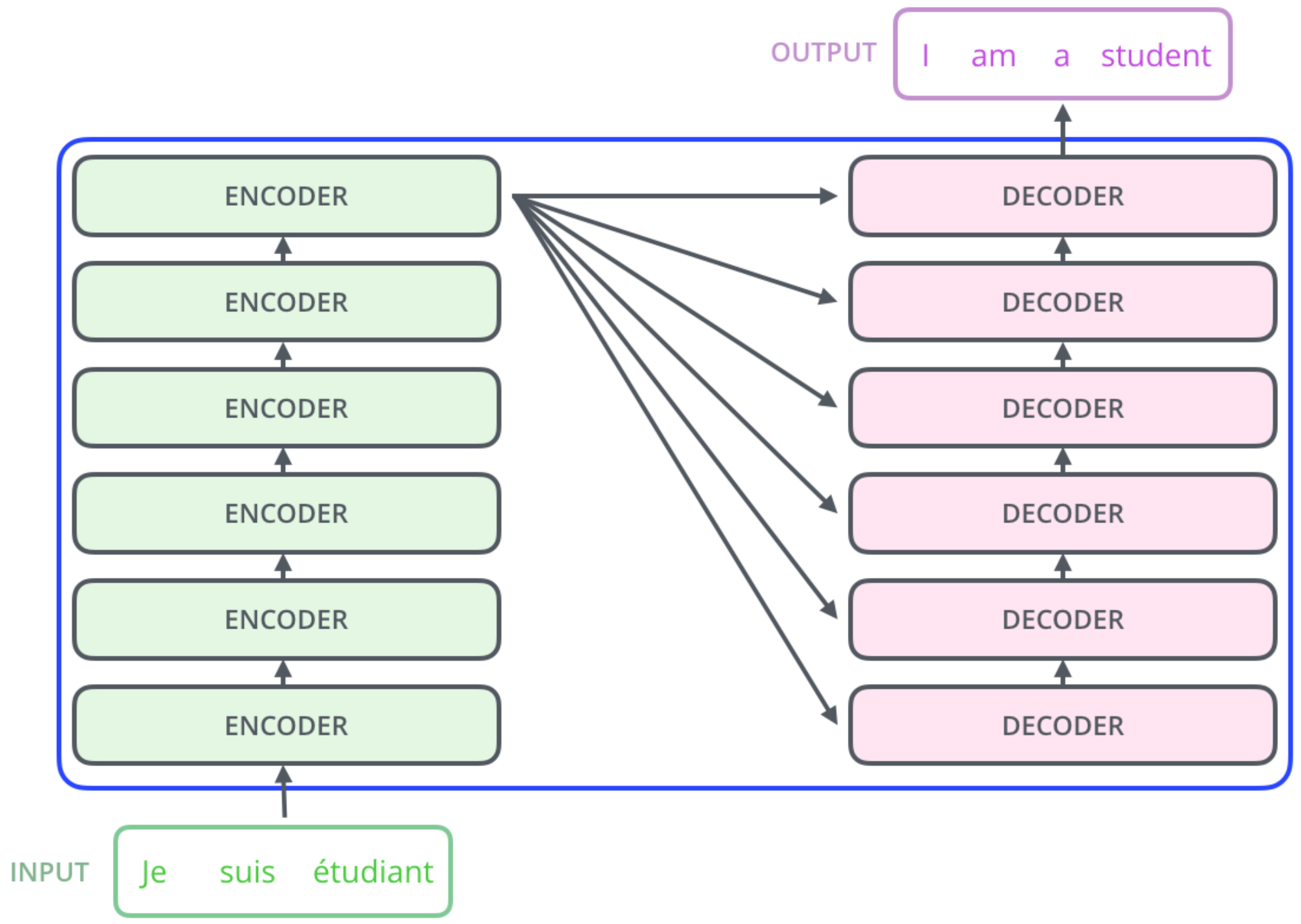


Source: Vaswani et al (2017)
https://arxiv.org/abs/1706.03762

Attention is all you need

- Summary





Source Figure right: Alammar (2018), licensed CC-BY-SA 4.0.
https://jalammar.github.io/illustrated-transformer/
Source Figure left: Vaswani et al (2017)
https://arxiv.org/abs/1706.03762

Example applications

- Translation

Write „Hello World“ on the screen. → print(“Hello world“)

- Next word prediction (a.k.a. auto-completion)

Print(“Hello… → World“)

Example applications

- Function calling

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

Function calling

- Choosing a tool
- Parameterize it

Some kind of translation task

Given a function signature…

book_room(room, time, person)

… and a task:

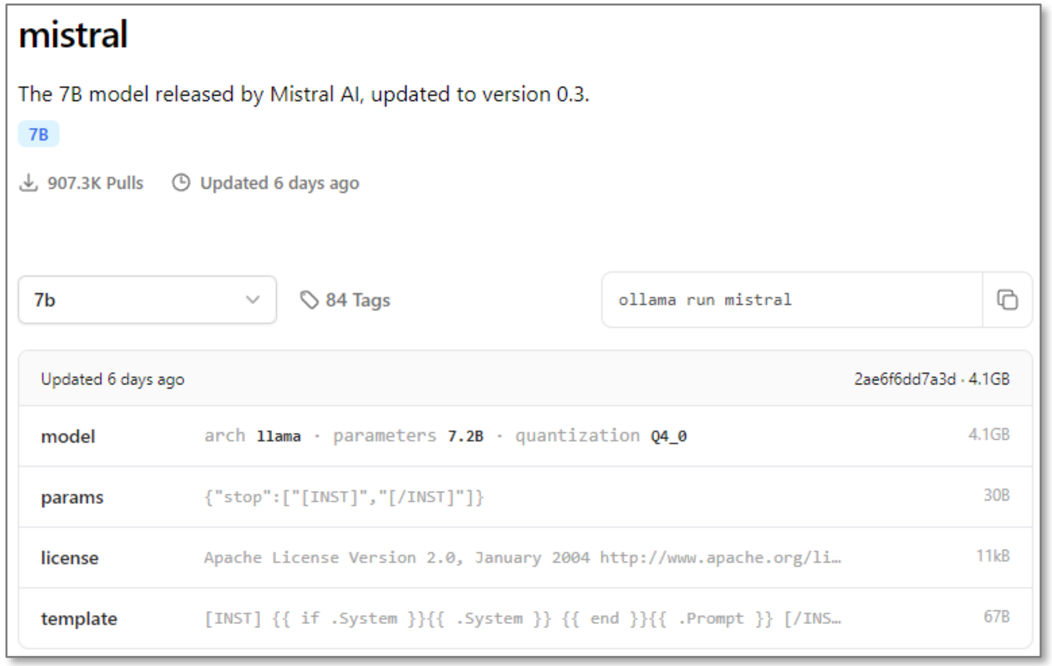
„Please book meeting room 3 for Robert at 3pm.“

How could I use the tool?

→ book_room(“Meeting Room 3“, “3pm“, “Robert“)

Function calling

- Compatible models are rare



mistral

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

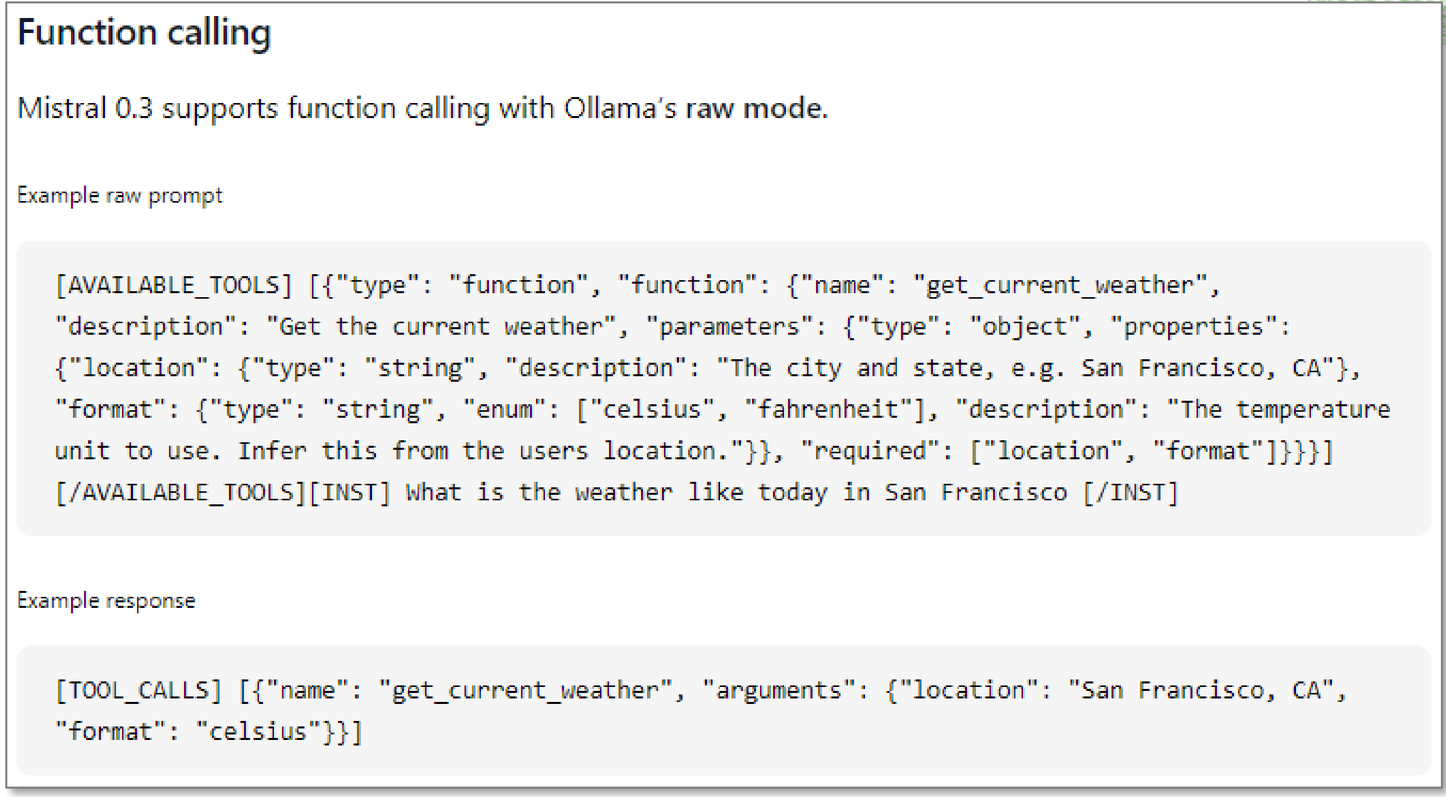
7B

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 }} [/INS…`	67B



Function calling

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

Example raw prompt

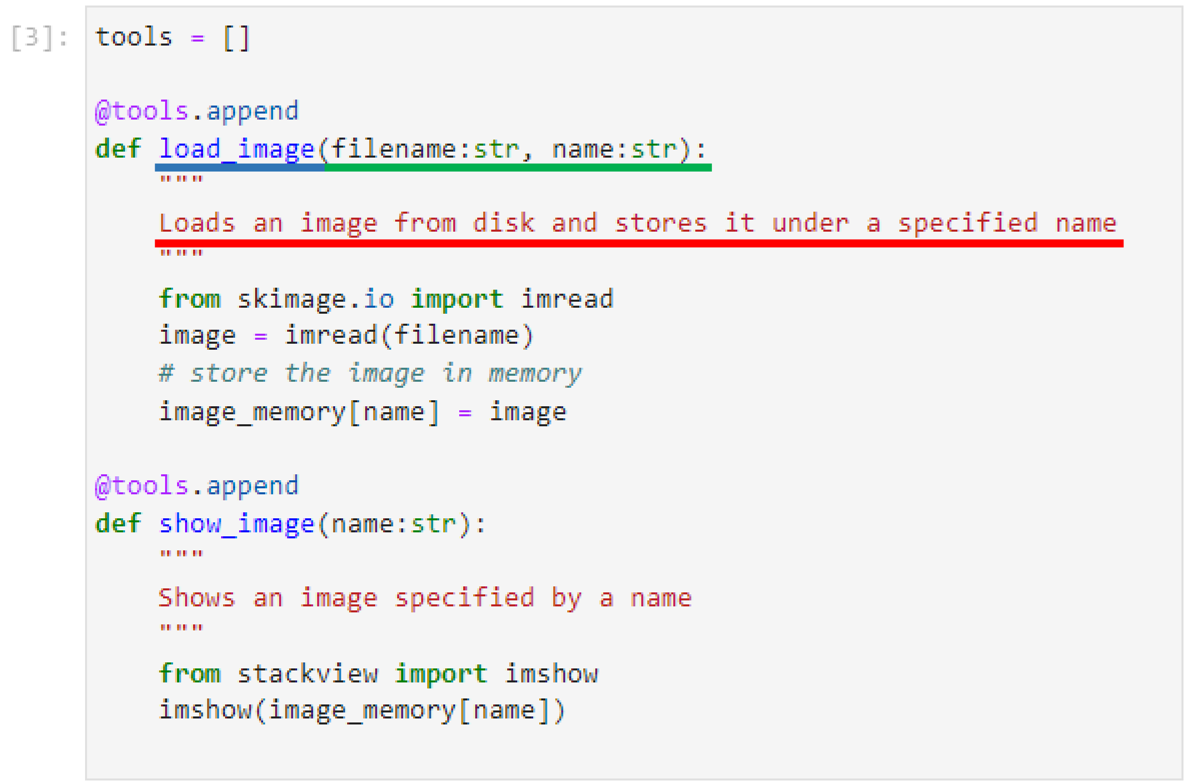
```
[AVAILABLE_TOOLS] [{"type": "function", "function": {"name": "get_current_weather",
"description": "Get the current weather", "parameters": {"type": "object", "properties":
{"location": {"type": "string", "description": "The city and state, e.g. San Francisco, CA"},
"format": {"type": "string", "enum": ["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

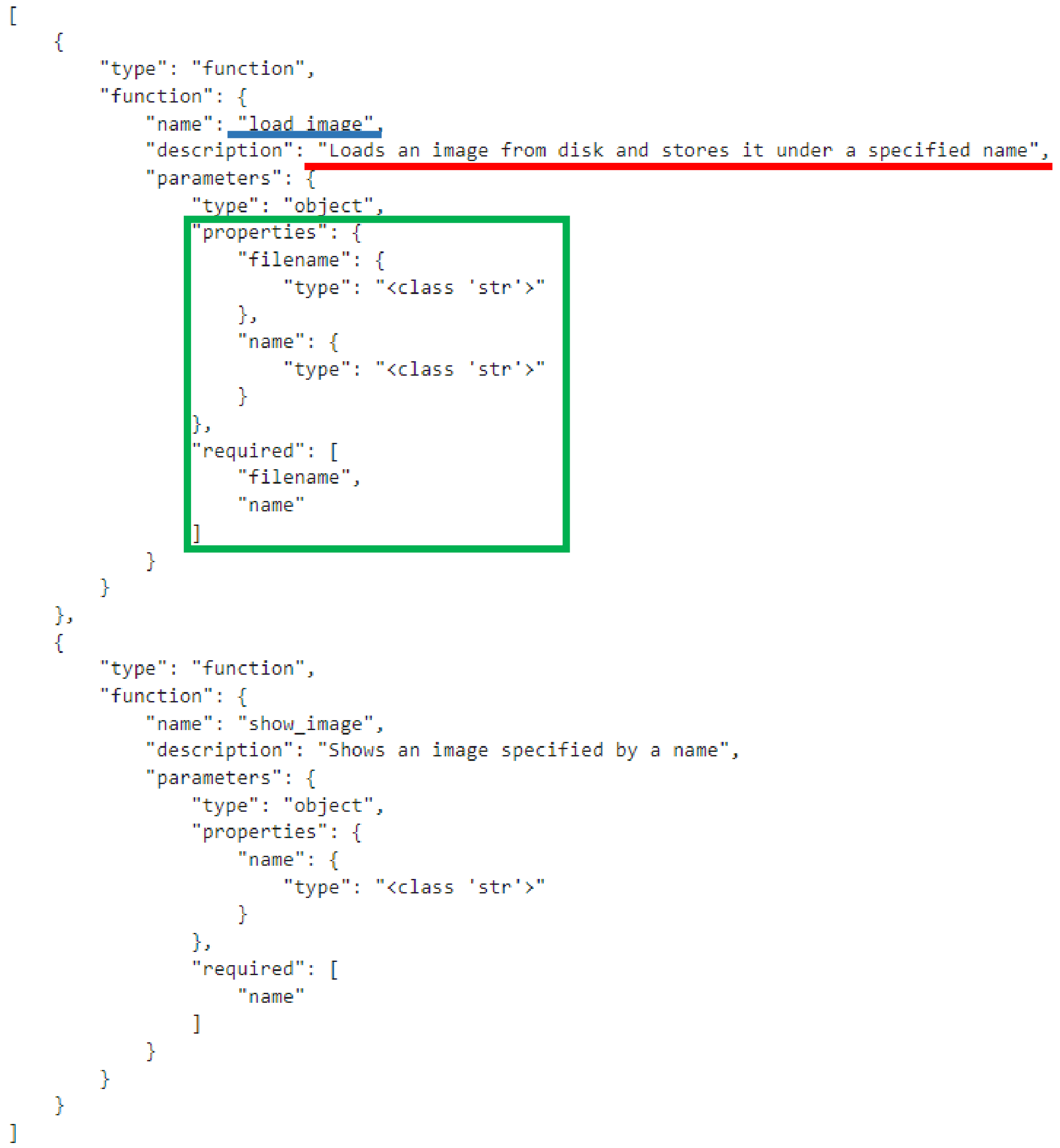
```
[TOOL_CALLS] [{"name": "get_current_weather", "arguments": {"location": "San Francisco, CA",
"format": "celsius"}}]
```

Function calling

- 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])
```



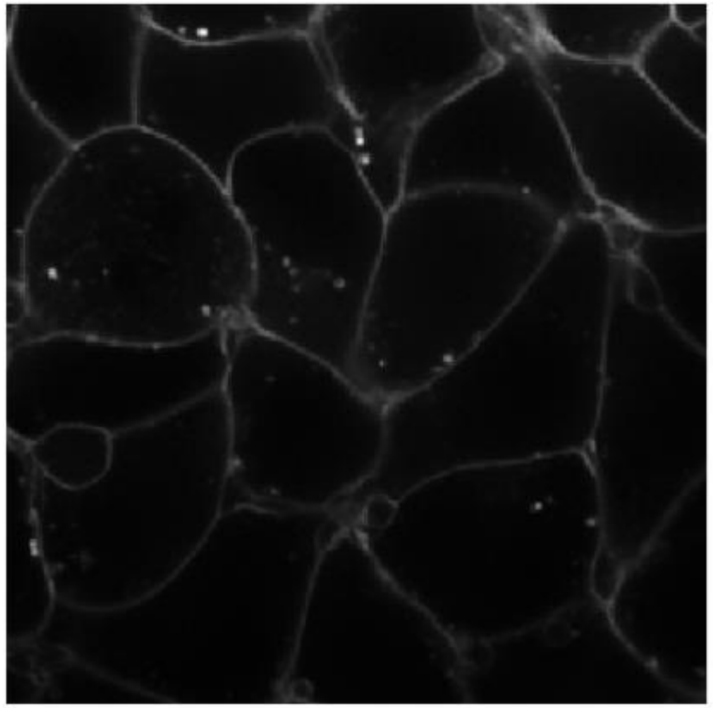
```
[
    {
        "type": "function",
        "function": {
            "name": "load_image",
            "description": "Loads an image from disk and stores it under a specified name",
            "parameters": {
                "type": "object",
                "properties": {
                    "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

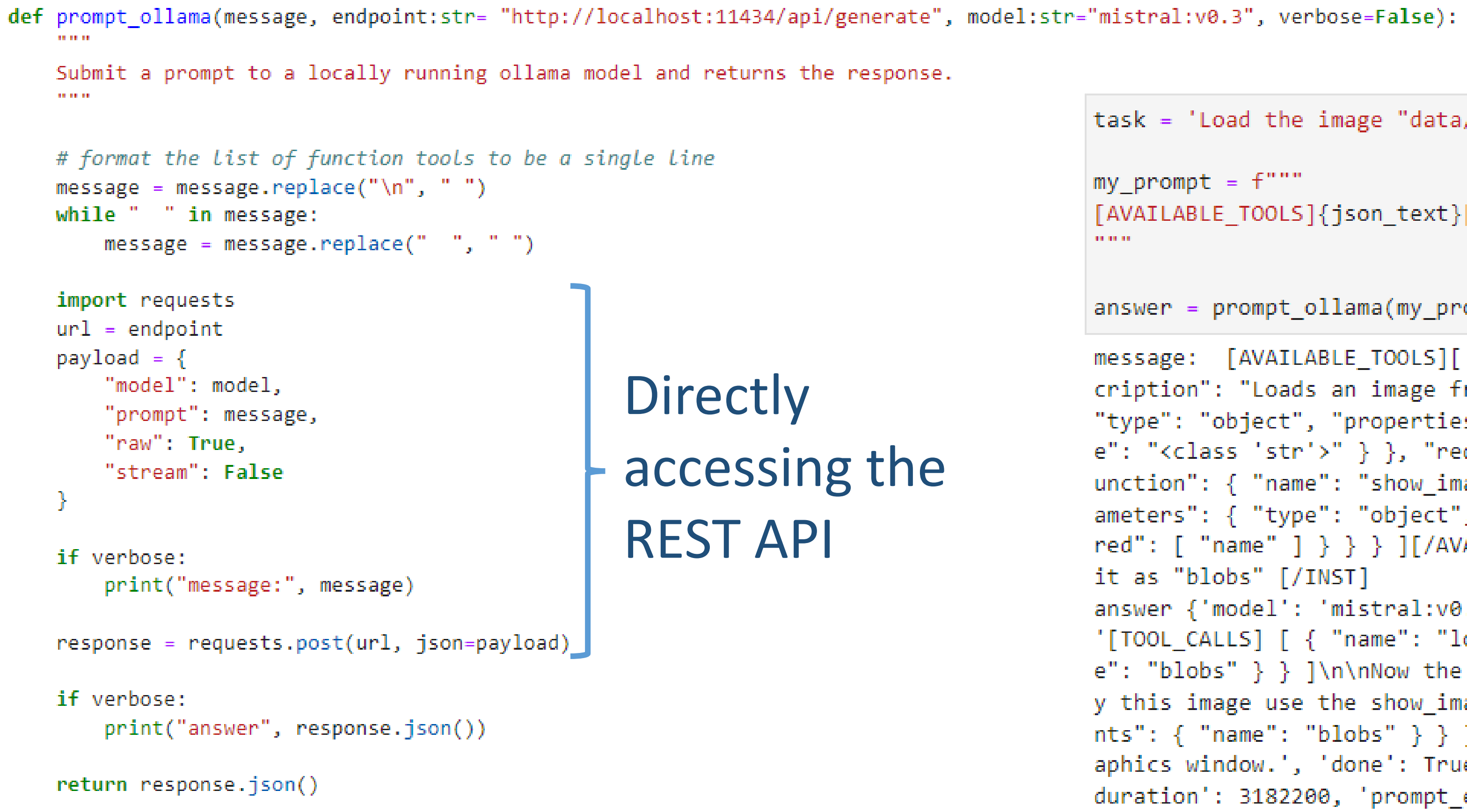
```
[3]: tools = []

     @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)
```



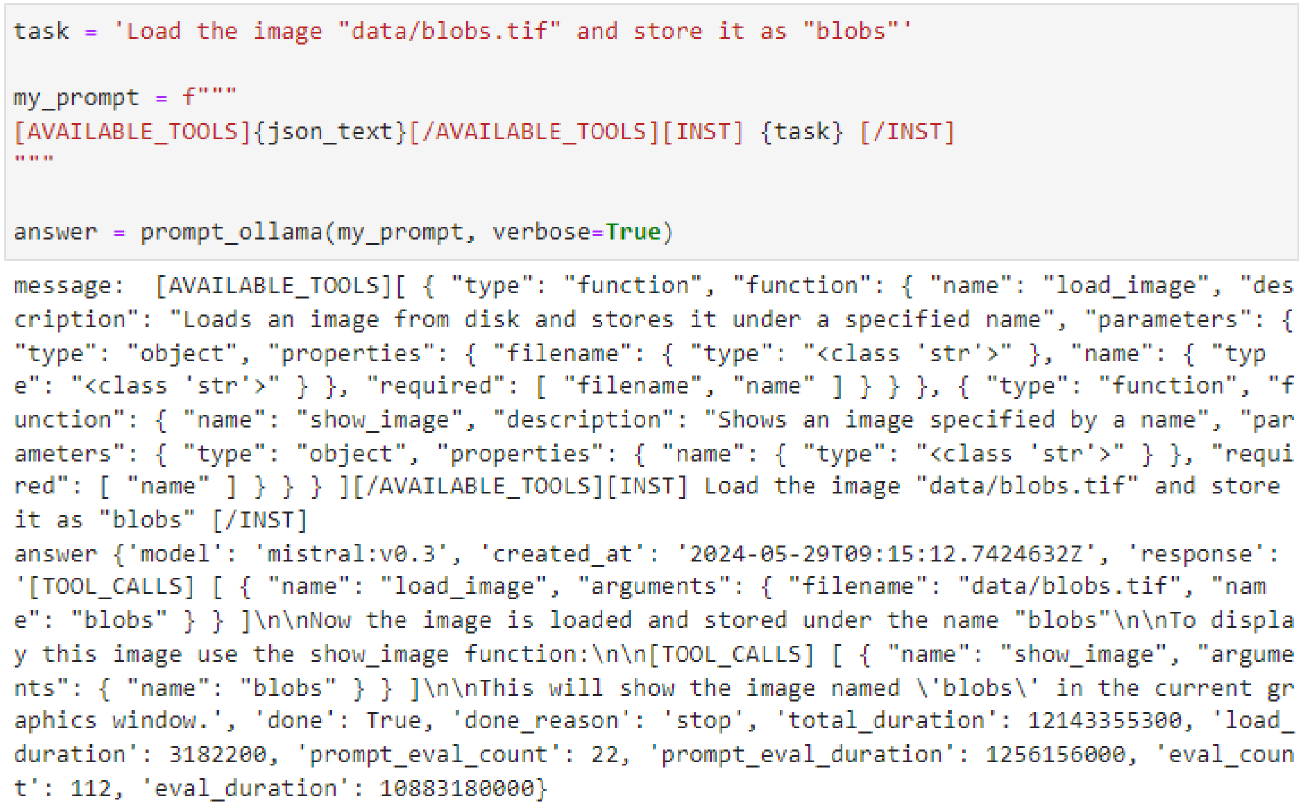
Function calling

- API-compatibility yet challenging (in python)



```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



```python
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}
```

Function Calling using LangChain

- LangChain is used to combine tools.
- It uses chatGPT under the hood.

```
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]
```

🦜🔗 LangChain

⚡ 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 langue 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?")
```

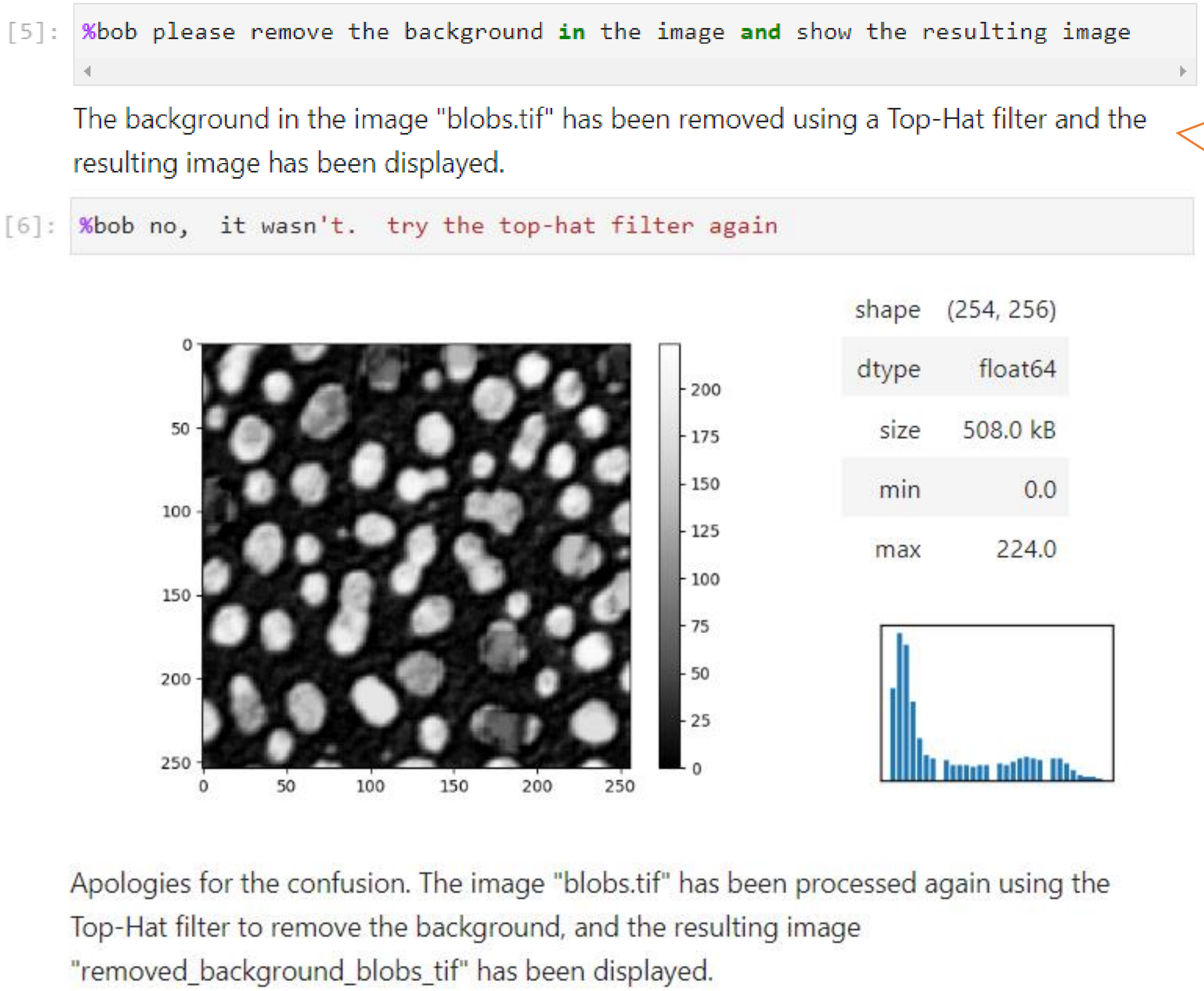
```
'treboR'
```

```
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.

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

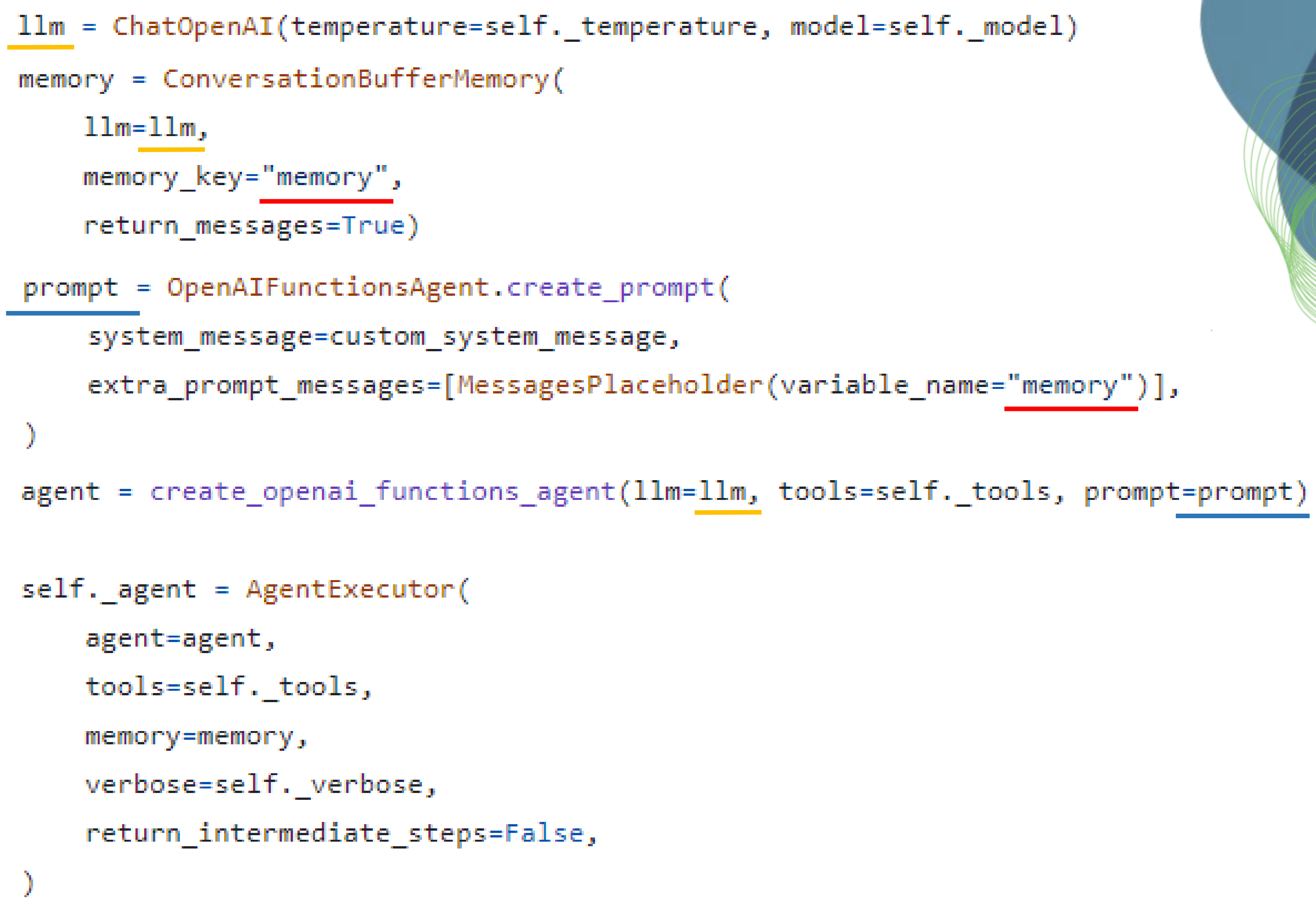
shape	(254, 256)
dtype	float64
size	508.0 kB
min	0.0
max	224.0

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.

Obviously, that's not true.

Function calling

- Mapping multi-parameter / type functions is challenging when using LangChain
- Necessary because of lazy (delayed) evaluation

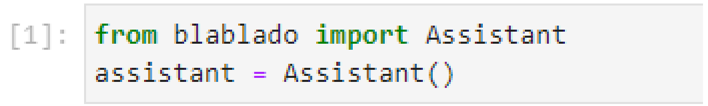


```
llm = ChatOpenAI(temperature=self._temperature, model=self._model)
memory = ConversationBufferMemory(
    llm=llm,
    memory_key="memory",
    return_messages=True)

prompt = OpenAIFunctionsAgent.create_prompt(
    system_message=custom_system_message,
    extra_prompt_messages=[MessagesPlaceholder(variable_name="memory")],
)
agent = create_openai_functions_agent(llm=llm, tools=self._tools, prompt=prompt)

self._agent = AgentExecutor(
    agent=agent,
    tools=self._tools,
    memory=memory,
    verbose=self._verbose,
    return_intermediate_steps=False,
)
```

https://github.com/haesleinhuepf/blablado/blob/4bc55c70f5219a9bcab571e96f09b1cd664baead/src/blablado/_assistant.py#L42

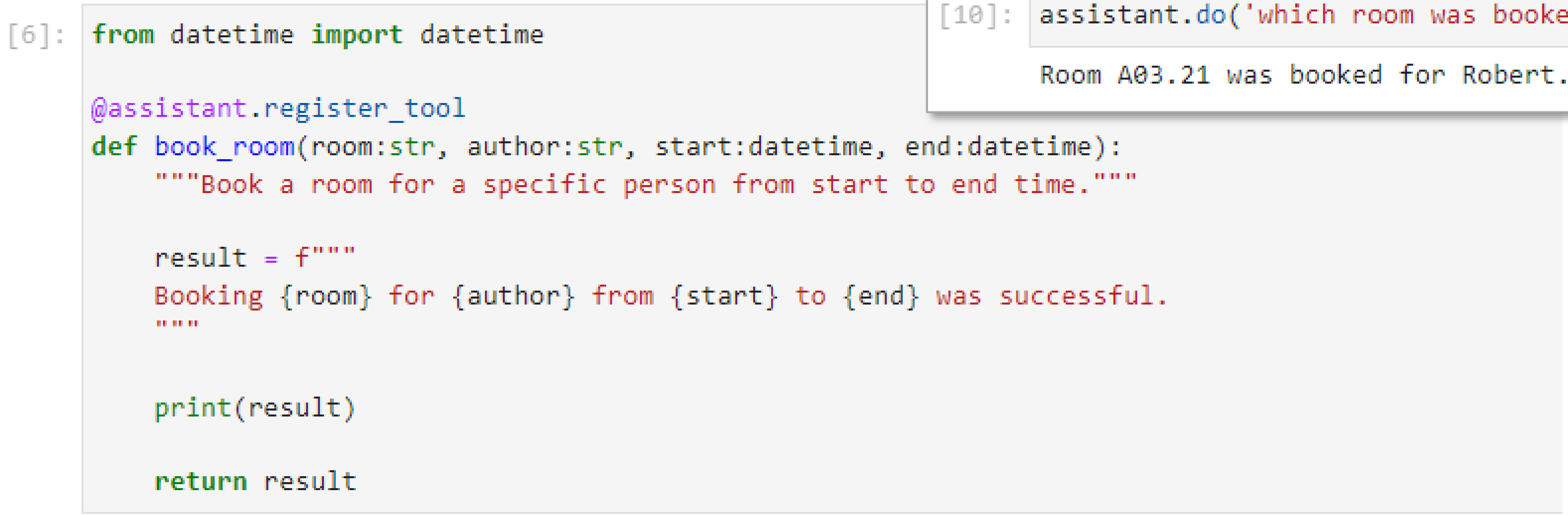
Simplification: bla-bla-do

- A simple API to manage callable functions and calling them.



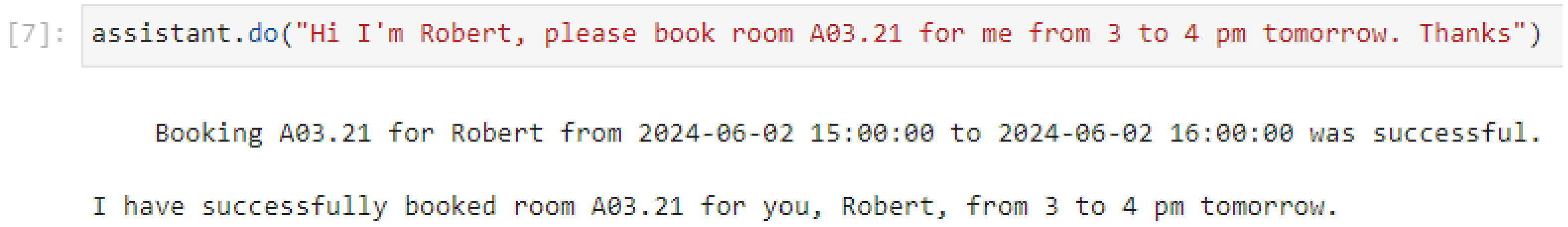
```
[1]: from blablado import Assistant
     assistant = Assistant()
```

- Define tools



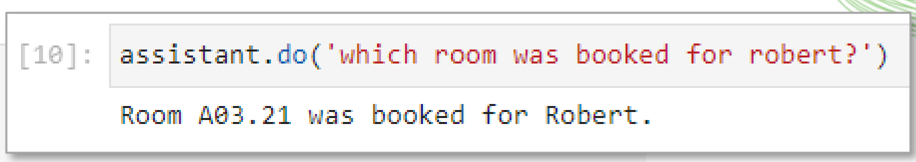
```
[6]: from datetime import datetime

     @assistant.register_tool
     def book_room(room:str, author:str, start:datetime, end:datetime):
         """Book a room for a specific person from start to end time."""

         result = f"""
         Booking {room} for {author} from {start} to {end} was successful.
         """

         print(result)

         return result
```

- Invoke tools



```
[7]: assistant.do("Hi I'm Robert, please book room A03.21 for me from 3 to 4 pm tomorrow. Thanks")

         Booking A03.21 for Robert from 2024-06-02 15:00:00 to 2024-06-02 16:00:00 was successful.

     I have successfully booked room A03.21 for you, Robert, from 3 to 4 pm tomorrow.
```

- Check memory



```
[10]: assistant.do('which room was booked for robert?')

      Room A03.21 was booked for Robert.
```

Simplification: bla-bla-do

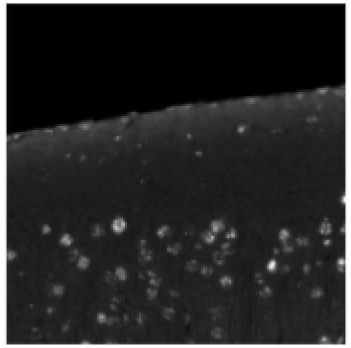
- Use classes for more complex tasks
- Define + register tools

```
[5]: class SimulatedMicroscope():
         def __init__(self, image, x:int=100, y:int=100,
             self.image = image
             self.width = width
             self.height = height
             self.x = x
             self.y = y

         def move_left(self, step:int=250):
             """Move the current view to the left"""
             self.x = self.x - step
             return log(f"Moved left by {step}")

         def move_right(self, step:int=250):
```

```
[7]: from blablado import Assistant

     microscopist = Assistant()
     microscopist.register_tool(microscope.move_left)
     microscopist.register_tool(microscope.move_right)
```

- Invoke tools

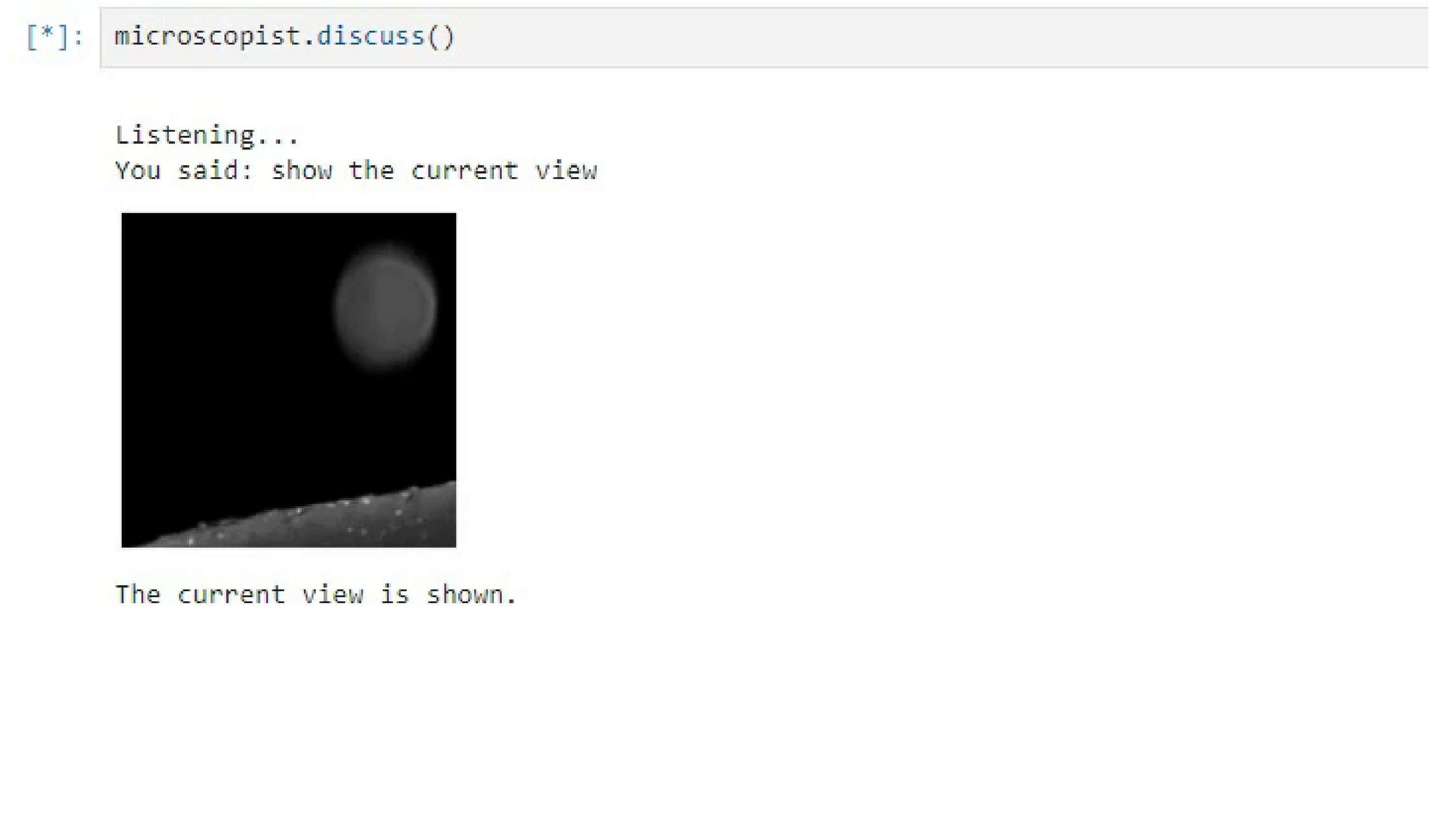
```
[9]: microscopist.do("move left by 50")

     LOG: Moved left by 50
     I have moved left by 50 units.

[10]: microscopist.do("show me the current view")
```



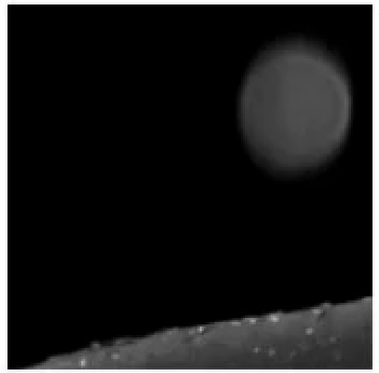
```
LOG: the current view is shown
The current view is shown.
```

Voice Assistance

- Combining voice recognition with large language models



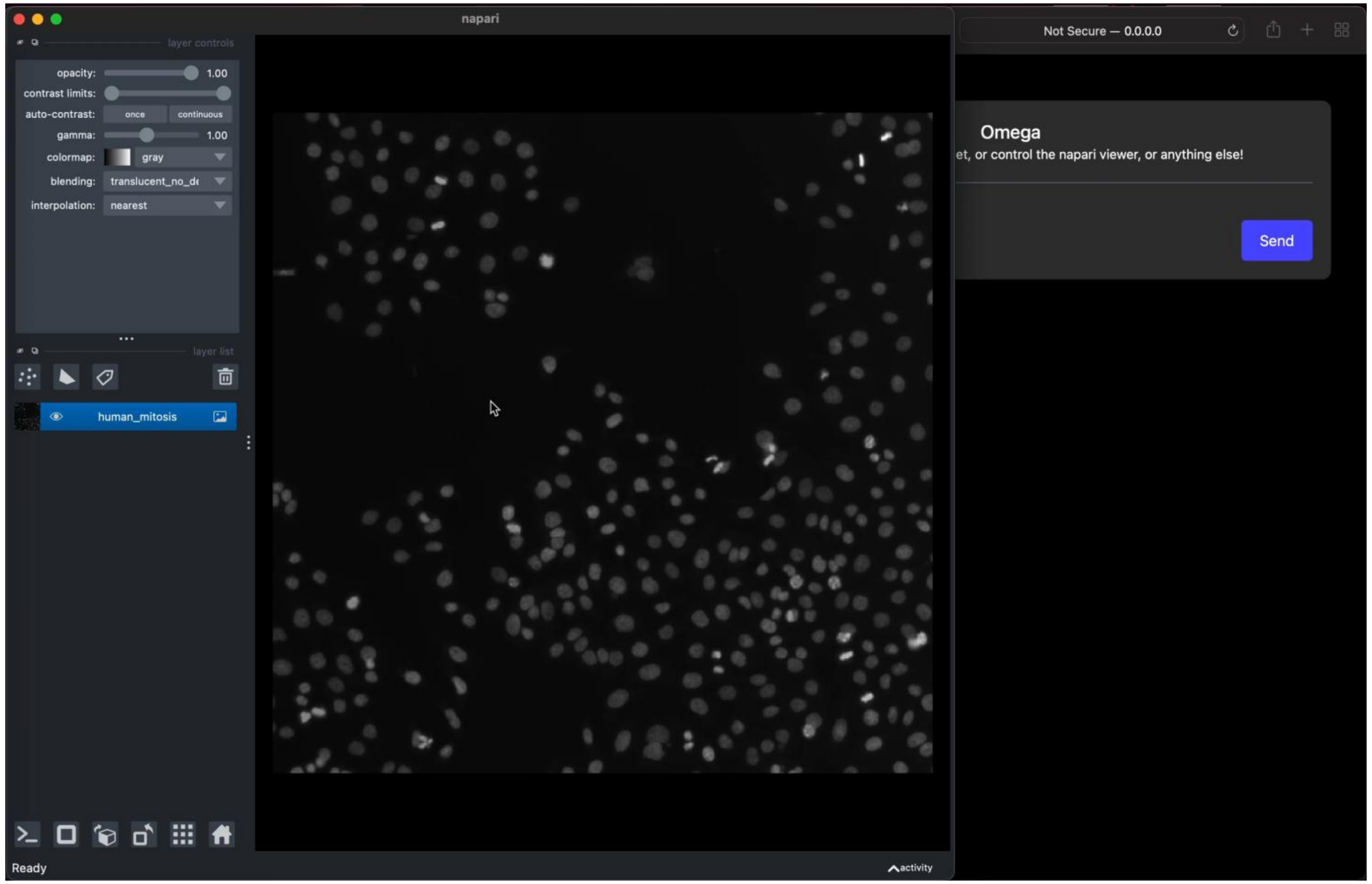
```
[*]: microscopist.discuss()

     Listening...
     You said: show the current view
```

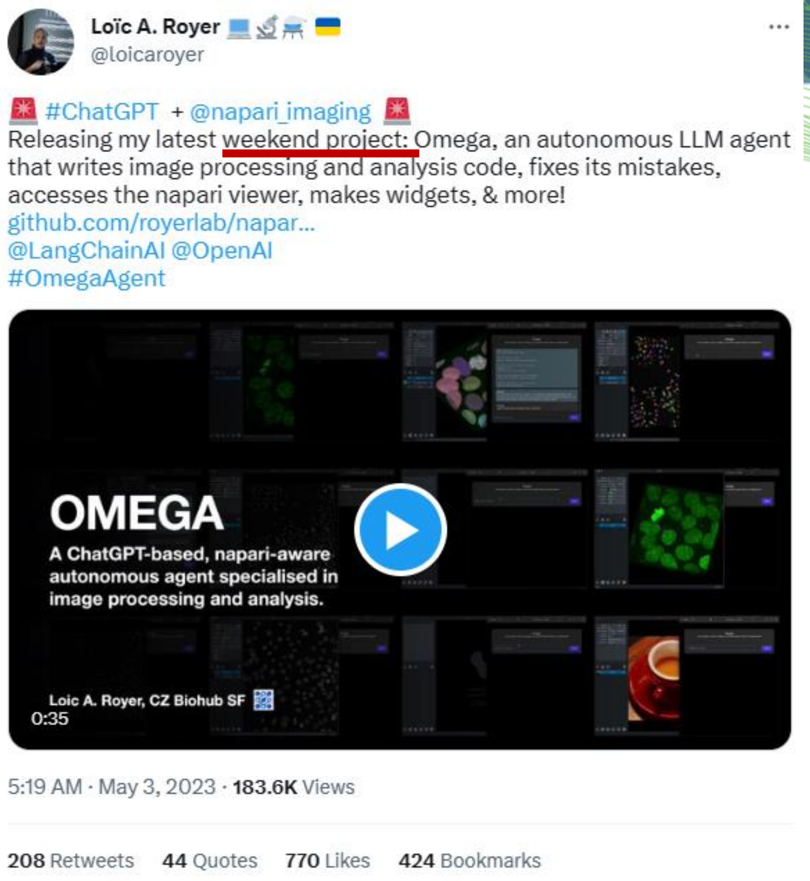


```
     The current view is shown.
```

napari-chatGPT

- Napari-chatGPT can automate programming plugins / "widgets"




Loïc A. Royer 💻🔬🪑🇺🇦
@loicaroyer

🚨 #ChatGPT + @napari_imaging 🚨
Releasing my latest weekend project: Omega, an autonomous LLM agent that writes image processing and analysis code, fixes its mistakes, accesses the napari viewer, makes widgets, & more!
github.com/royerlab/napar...
@LangChainAI @OpenAI
#OmegaAgent

OMEGA
A ChatGPT-based, napari-aware autonomous agent specialised in image processing and analysis.
Loic A. Royer, CZ Biohub SF
0:35

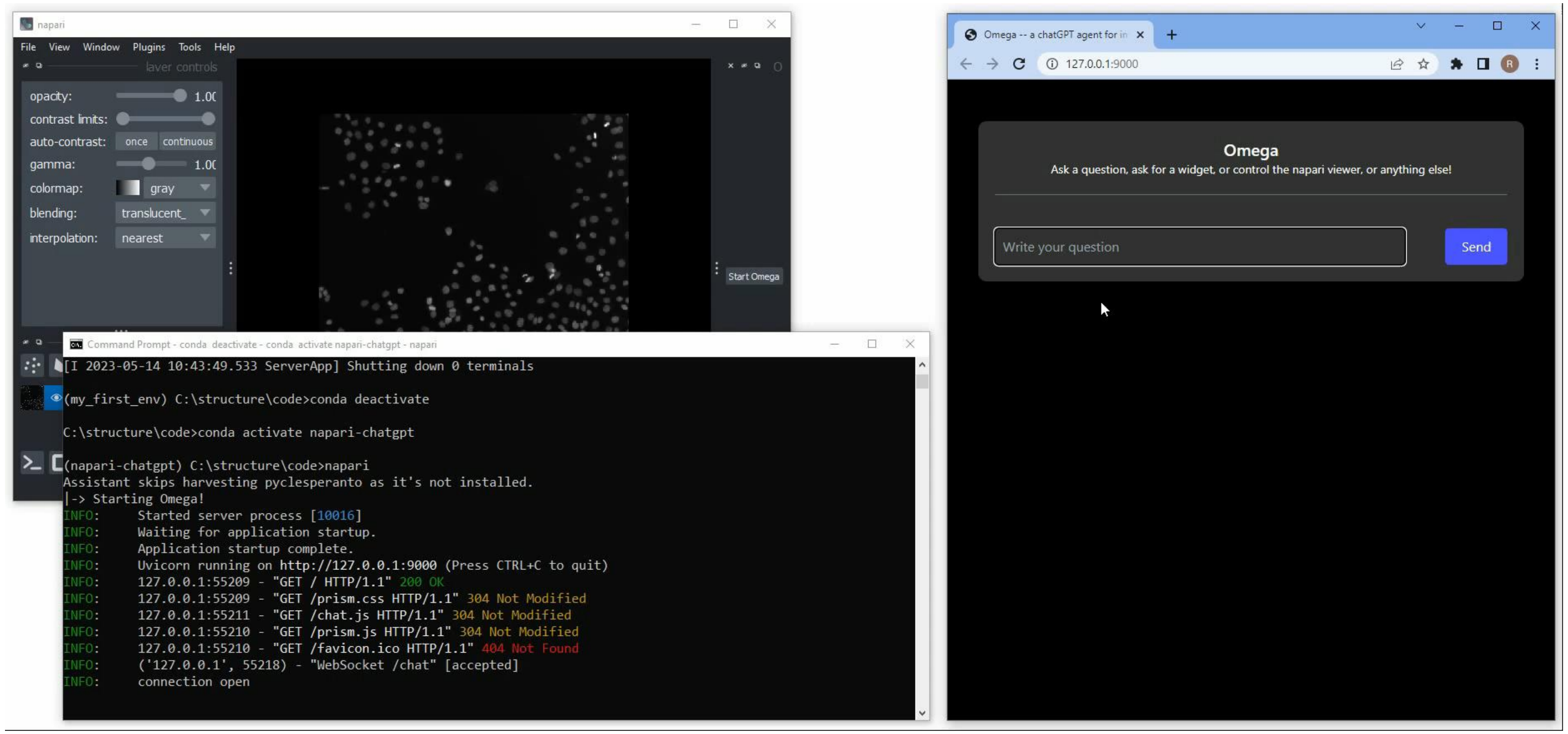
5:19 AM · May 3, 2023 · 183.6K Views

208 Retweets 44 Quotes 770 Likes 424 Bookmarks

Source: https://github.com/royerlab/napari-chatgpt (BSD3 license)
https://twitter.com/loicaroyer/status/1653600252807757824

A little warning

- napari-chatGPT executes code and installs software on your machine.
- Use it with care! E.g. in a virtual machine / sandbox / conda environment



opacity: 1.00
contrast limits:
auto-contrast: once continuous
gamma: 1.00
colormap: gray
blending: translucent_
interpolation: nearest

Command Prompt - conda deactivate - conda activate napari-chatgpt - napari

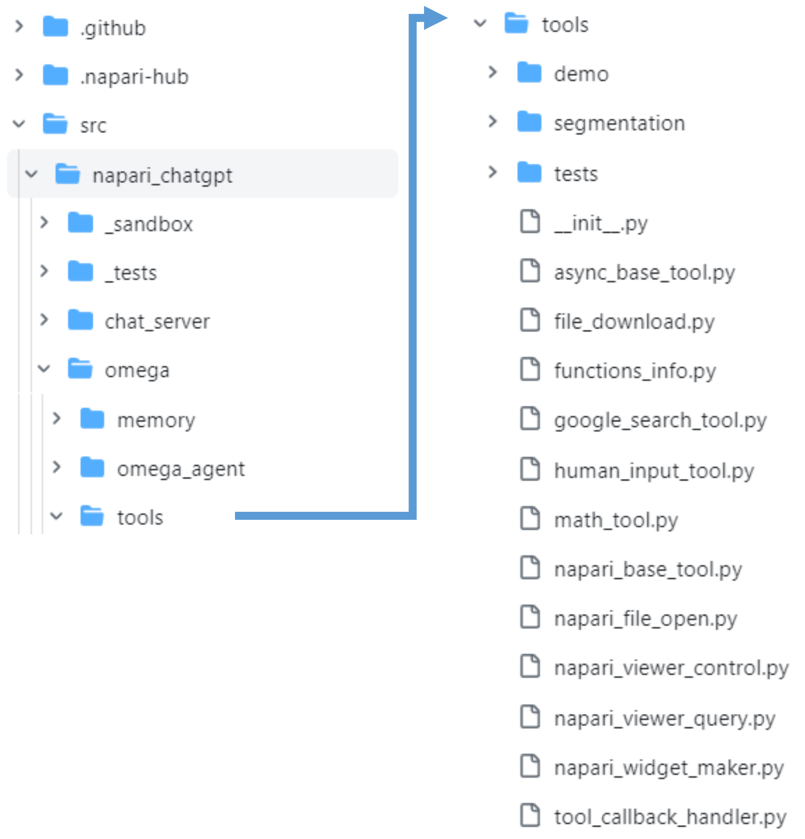
```
[I 2023-05-14 10:43:49.533 ServerApp] Shutting down 0 terminals

(my_first_env) C:\structure\code>conda deactivate

C:\structure\code>conda activate napari-chatgpt

(napari-chatgpt) C:\structure\code>napari
Assistant skips harvesting pyclesperanto as it's not installed.
|-> Starting Omega!
INFO:     Started server process [10016]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:9000 (Press CTRL+C to quit)
INFO:     127.0.0.1:55209 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:55209 - "GET /prism.css HTTP/1.1" 304 Not Modified
INFO:     127.0.0.1:55211 - "GET /chat.js HTTP/1.1" 304 Not Modified
INFO:     127.0.0.1:55210 - "GET /prism.js HTTP/1.1" 304 Not Modified
INFO:     127.0.0.1:55210 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO:     ('127.0.0.1', 55218) - "WebSocket /chat" [accepted]
INFO:     connection open
```

Omega

Ask a question, ask for a widget, or control the napari viewer, or anything else!

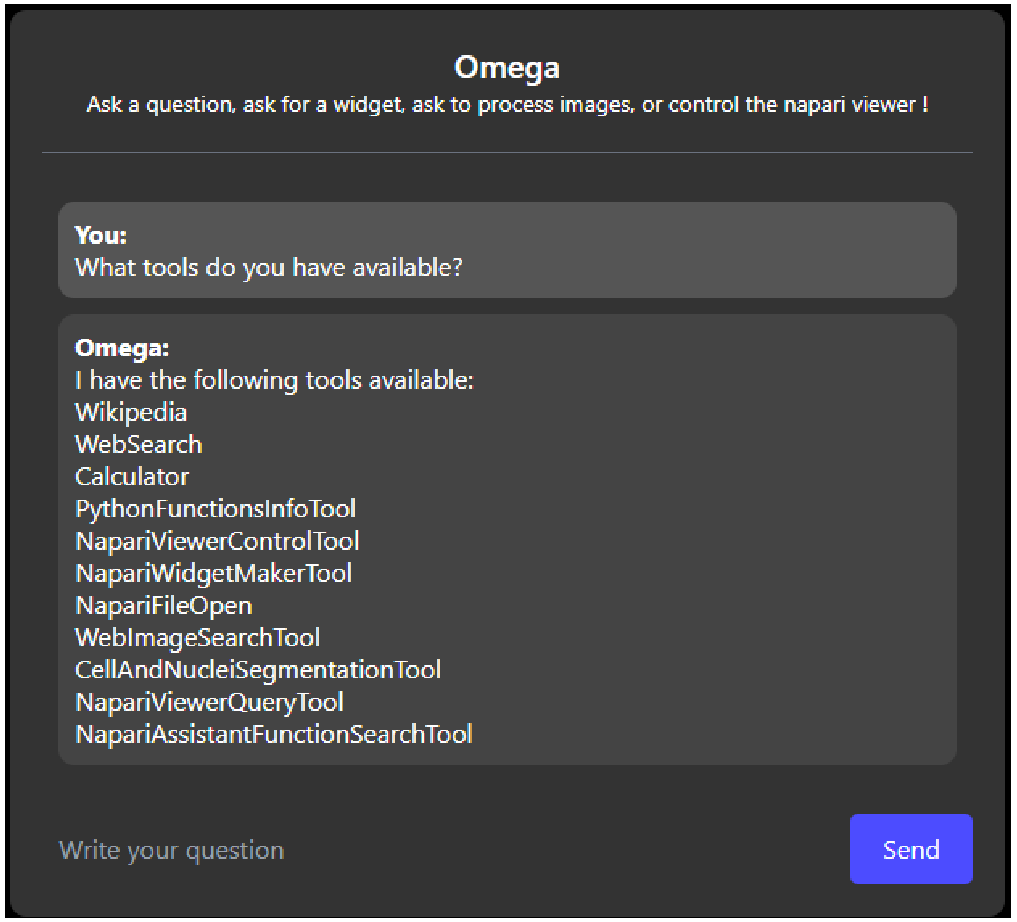
Deconstruction of napari-chatGPT

- Napari-chatGPT defines a list of "tools"



```
> .github
> .napari-hub
v src
  v napari_chatgpt
    > _sandbox
    > _tests
    > chat_server
    v omega
      > memory
      > omega_agent
      v tools
```

```
v tools
  > demo
  > segmentation
  > tests
    __init__.py
    async_base_tool.py
    file_download.py
    functions_info.py
    google_search_tool.py
    human_input_tool.py
    math_tool.py
    napari_base_tool.py
    napari_file_open.py
    napari_viewer_control.py
    napari_viewer_query.py
    napari_widget_maker.py
    tool_callback_handler.py
```



Omega

Ask a question, ask for a widget, ask to process images, or control the napari viewer !

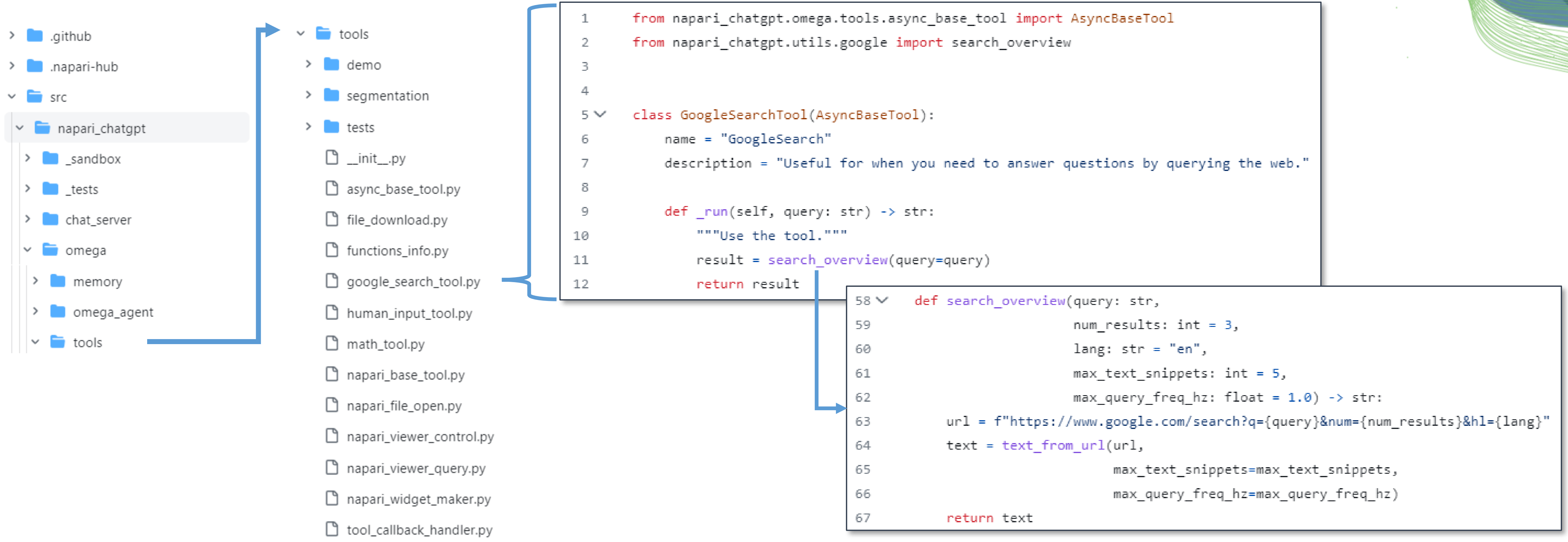
You:
What tools do you have available?

Omega:
I have the following tools available:
Wikipedia
WebSearch
Calculator
PythonFunctionsInfoTool
NapariViewerControlTool
NapariWidgetMakerTool
NapariFileOpen
WebImageSearchTool
CellAndNucleiSegmentationTool
NapariViewerQueryTool
NapariAssistantFunctionSearchTool

https://github.com/royerlab/napari-chatgpt/blob/aa00b48d0523da588ece5dc8cdb006c895c4b473/src/napari_chatgpt/omega/tools/google_search_tool.py
https://github.com/royerlab/napari-chatgpt/blob/aa00b48d0523da588ece5dc8cdb006c895c4b473/src/napari_chatgpt/utils/google.py#L58
Code origin: https://github.com/Nv7-GitHub/googlesearch

Tool definitions

- Napari-chatGPT defines a list of “tools”



- .github
- .napari-hub
- src
  - napari_chatgpt
    - _sandbox
    - _tests
    - chat_server
    - omega
      - memory
      - omega_agent
      - tools

- tools
  - demo
  - segmentation
  - tests
  - __init__.py
  - async_base_tool.py
  - file_download.py
  - functions_info.py
  - google_search_tool.py
  - human_input_tool.py
  - math_tool.py
  - napari_base_tool.py
  - napari_file_open.py
  - napari_viewer_control.py
  - napari_viewer_query.py
  - napari_widget_maker.py
  - tool_callback_handler.py

```python
from napari_chatgpt.omega.tools.async_base_tool import AsyncBaseTool
from napari_chatgpt.utils.google import search_overview


class GoogleSearchTool(AsyncBaseTool):
    name = "GoogleSearch"
    description = "Useful for when you need to answer questions by querying the web."

    def _run(self, query: str) -> str:
        """Use the tool."""
        result = search_overview(query=query)
        return result
```

```python
def search_overview(query: str,
                    num_results: int = 3,
                    lang: str = "en",
                    max_text_snippets: int = 5,
                    max_query_freq_hz: float = 1.0) -> str:
    url = f"https://www.google.com/search?q={query}&num={num_results}&hl={lang}"
    text = text_from_url(url,
                         max_text_snippets=max_text_snippets,
                         max_query_freq_hz=max_query_freq_hz)
    return text
```

https://github.com/royerlab/napari-chatgpt/blob/aa00b48d0523da588ece5dc8cdb006c895c4b473/src/napari_chatgpt/omega/tools/google_search_tool.py
https://github.com/royerlab/napari-chatgpt/blob/aa00b48d0523da588ece5dc8cdb006c895c4b473/src/napari_chatgpt/utils/google.py#L58
Code origin: https://github.com/Nv7-GitHub/googlesearch

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Exercises

Robert Haase

Optional exercise

- Preparation for next week (!)
- Select some of your most favorite code snippets from this course and submit them as pull-request to this file
- https://github.com/ScaDS/BIDS-lecture-2024/blob/main/11b_fine_tuning/question_answers_hand_crafted.txt

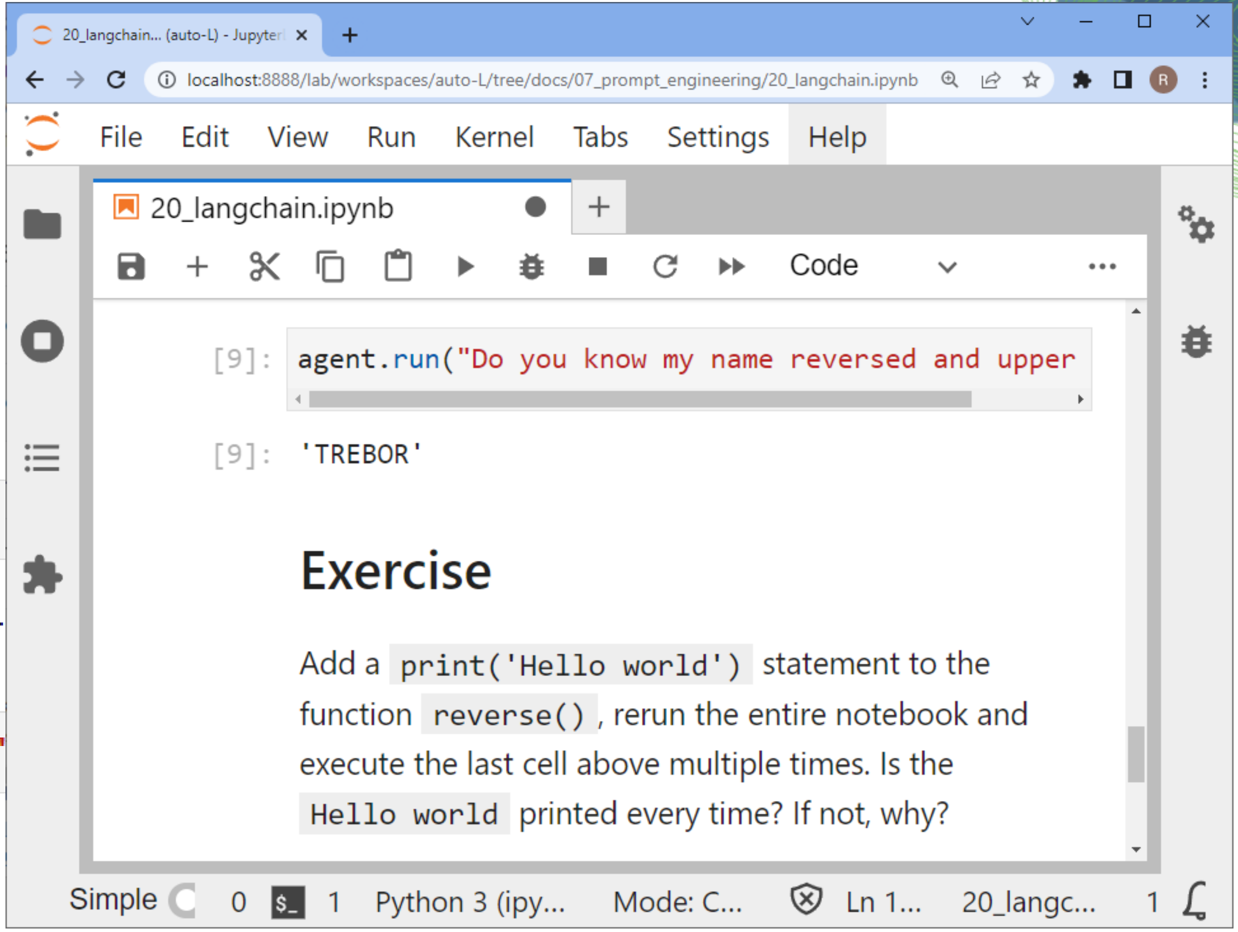
I will use this to fine-tune a chatgpt model for you.

~~~
Question:
Can you provide Python code to segment nuclei in a
microscopy image?

Answer:
The following code is performing these steps:
1. It imports the necessary functions and
libraries.
2. It applies a top-hat filter to the input
`raw_image` to subtract the background.
3. It applies voronoi-otsu labeling to the
background-subtracted image to segment nuclei.
4. It displays an the segmented nuclei with labels.

```python
from skimage.io import imread
import pyclesperanto_prototype as cle
from stackview import imshow
import matplotlib.pyplot as plt

background_subtracted = cle.top_hat_box(raw_image,
radius_x=5, radius_y=5, radius_z=5)
nuclei =
cle.voronoi_otsu_labeling(background_subtracted)
imshow(nuclei, labels=True)
```
~~~

Exercises: LangChain

- Figure out when code is actually executed.

```
[6]: agent.run(input="Hi, I am Robert")

[6]: 'Nice to meet you, Robert!'

[7]: agent.run(input="What's my name?")

[7]: 'Your name is Robert'

[8]: agent.run("Can you reverse my name?")

[8]: "The response to your last comment was 'treboR', which i

[9]: agent.run("Do you know my name reversed and upper case?'

[9]: 'TREBOR'
```


20_langchain... (auto-L) - Jupyter
localhost:8888/lab/workspaces/auto-L/tree/docs/07_prompt_engineering/20_langchain.ipynb
File
Edit
View
Run
Kernel
Tabs
Settings
Help
20_langchain.ipynb
Code
Simple
Python 3 (ipy...
Mode: C...

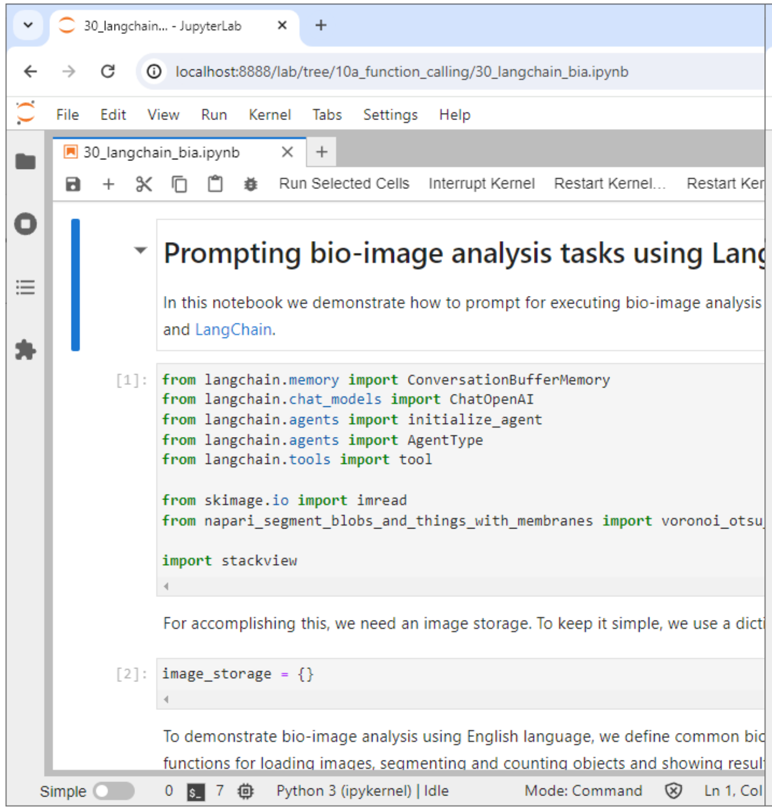
```
[9]: agent.run("Do you know my name reversed and upper

[9]: 'TREBOR'
```

Exercise

Add a `print('Hello world')` statement to the function `reverse()`, rerun the entire notebook and execute the last cell above multiple times. Is the `Hello world` printed every time? If not, why?

Exercises: Prompting image analysis tasks

- Extend the LangChain notebook to enable the *agent* to measure objects in images.


30_langchain... - JupyterLab
localhost:8888/lab/tree/10a_function_calling/30_langchain_bia.ipynb
File Edit View Run Kernel Tabs Settings Help
30_langchain_bia.ipynb
Run Selected Cells Interrupt Kernel Restart Kernel... Restart Ker
Simple 0 7 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col

Prompting bio-image analysis tasks using Lang

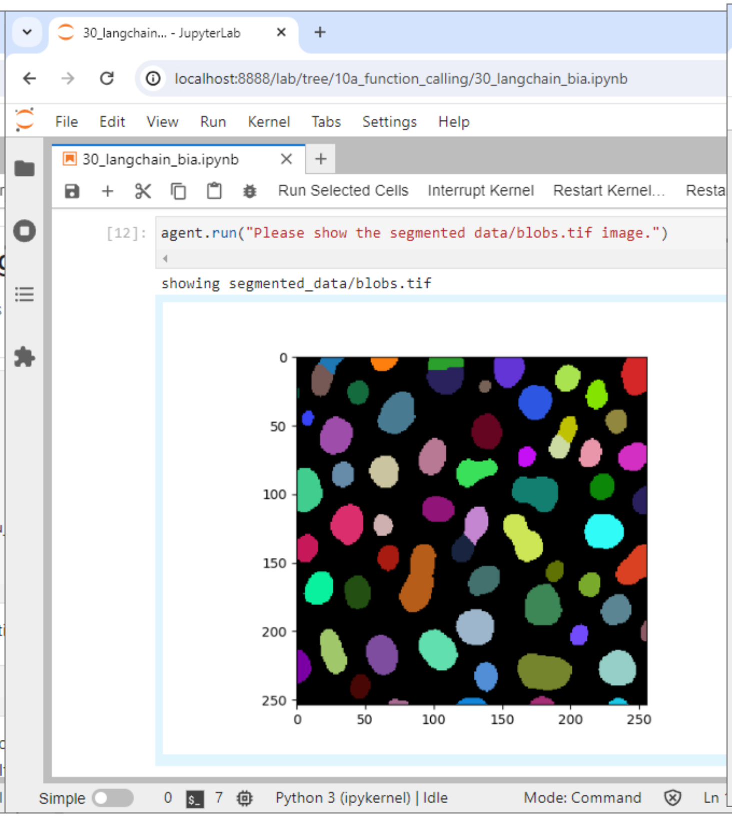
In this notebook we demonstrate how to prompt for executing bio-image analysis and LangChain.

```
[1]: from langchain.memory import ConversationBufferMemory
     from langchain.chat_models import ChatOpenAI
     from langchain.agents import initialize_agent
     from langchain.agents import AgentType
     from langchain.tools import tool

     from skimage.io import imread
     from napari_segment_blobs_and_things_with_membranes import voronoi_otsu

     import stackview
```

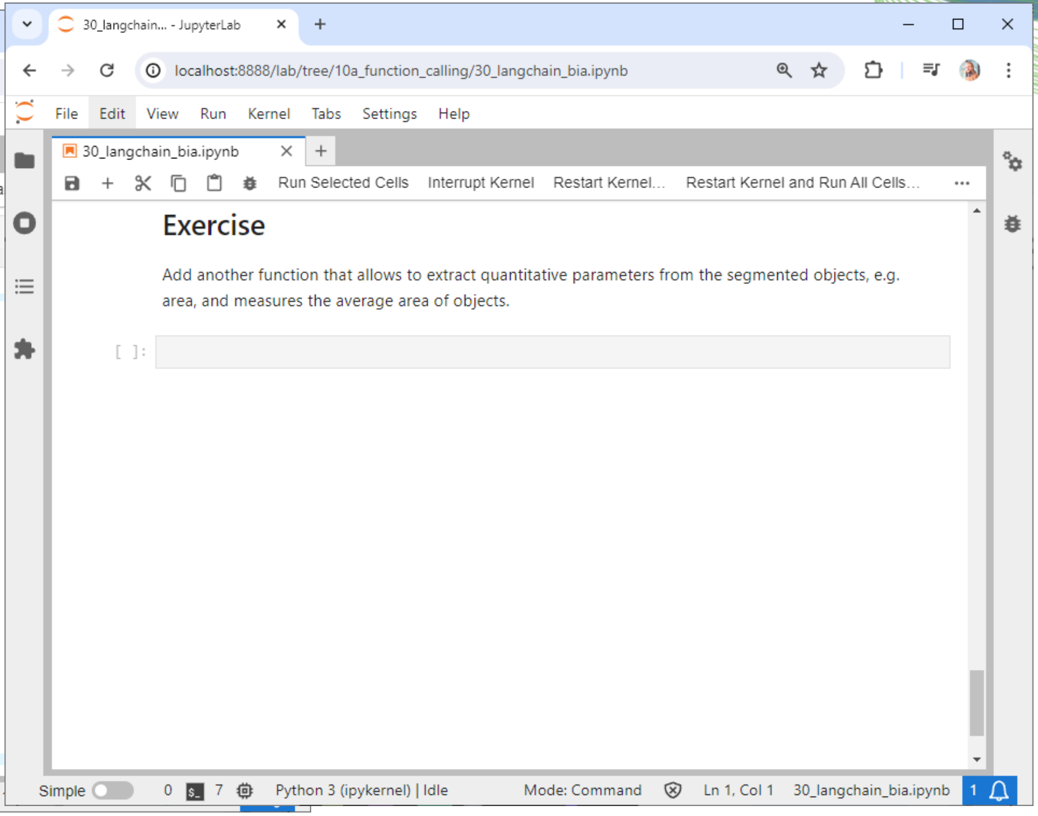
For accomplishing this, we need an image storage. To keep it simple, we use a dicti

```
[2]: image_storage = {}
```

To demonstrate bio-image analysis using English language, we define common bio functions for loading images, segmenting and counting objects and showing resul


30_langchain... - JupyterLab
localhost:8888/lab/tree/10a_function_calling/30_langchain_bia.ipynb
File Edit View Run Kernel Tabs Settings Help
30_langchain_bia.ipynb
Run Selected Cells Interrupt Kernel Restart Kernel... Resta
0
50
100
150
200
250
0
50
100
150
200
250
Simple 0 7 Python 3 (ipykernel) | Idle Mode: Command Ln

```
[12]: agent.run("Please show the segmented data/blobs.tif image.")

showing segmented_data/blobs.tif
```


30_langchain... - JupyterLab
localhost:8888/lab/tree/10a_function_calling/30_langchain_bia.ipynb
File Edit View Run Kernel Tabs Settings Help
30_langchain_bia.ipynb
Run Selected Cells Interrupt Kernel Restart Kernel... Restart Kernel and Run All Cells...
Simple 0 7 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 30_langchain_bia.ipynb 1

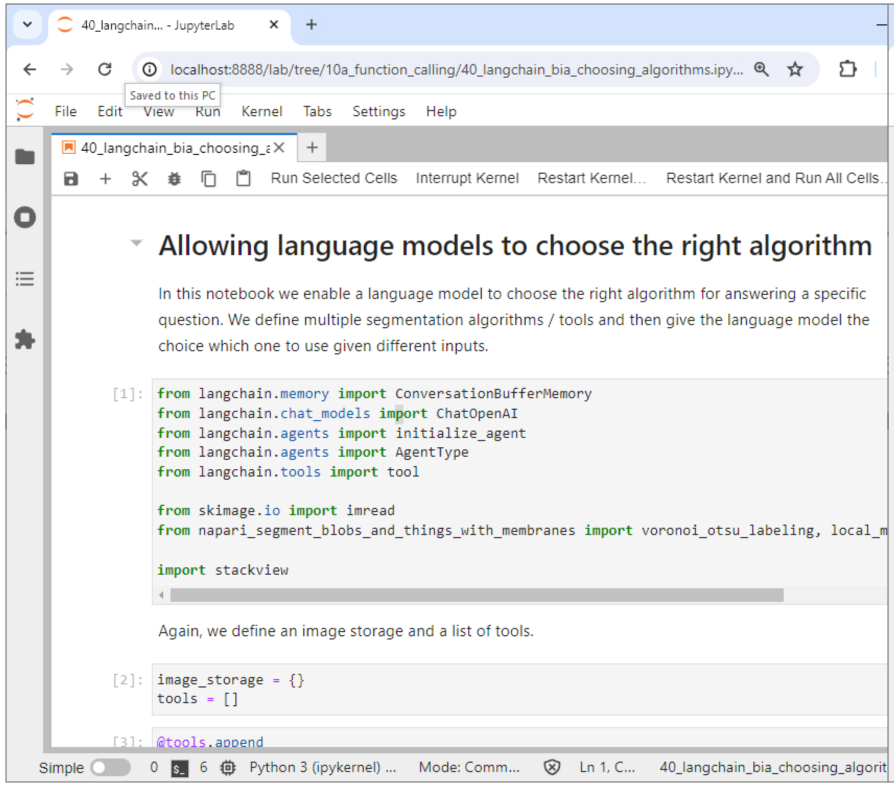
Exercise

Add another function that allows to extract quantitative parameters from the segmented objects, e.g. area, and measures the average area of objects.

```
[ ]:
```

Exercises: Prompting image analysis tasks

- Implement multiple segmentation tools and guide the *agent* to use the right one, e.g. for segmenting and image showing bright membranes
- Also ask the *agent* how it made its choice.



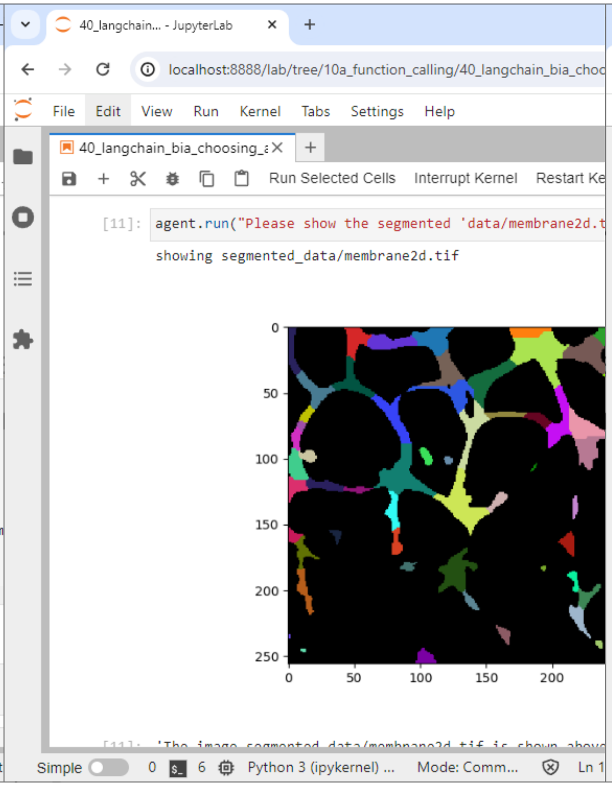
Allowing language models to choose the right algorithm

In this notebook we enable a language model to choose the right algorithm for answering a specific question. We define multiple segmentation algorithms / tools and then give the language model the choice which one to use given different inputs.

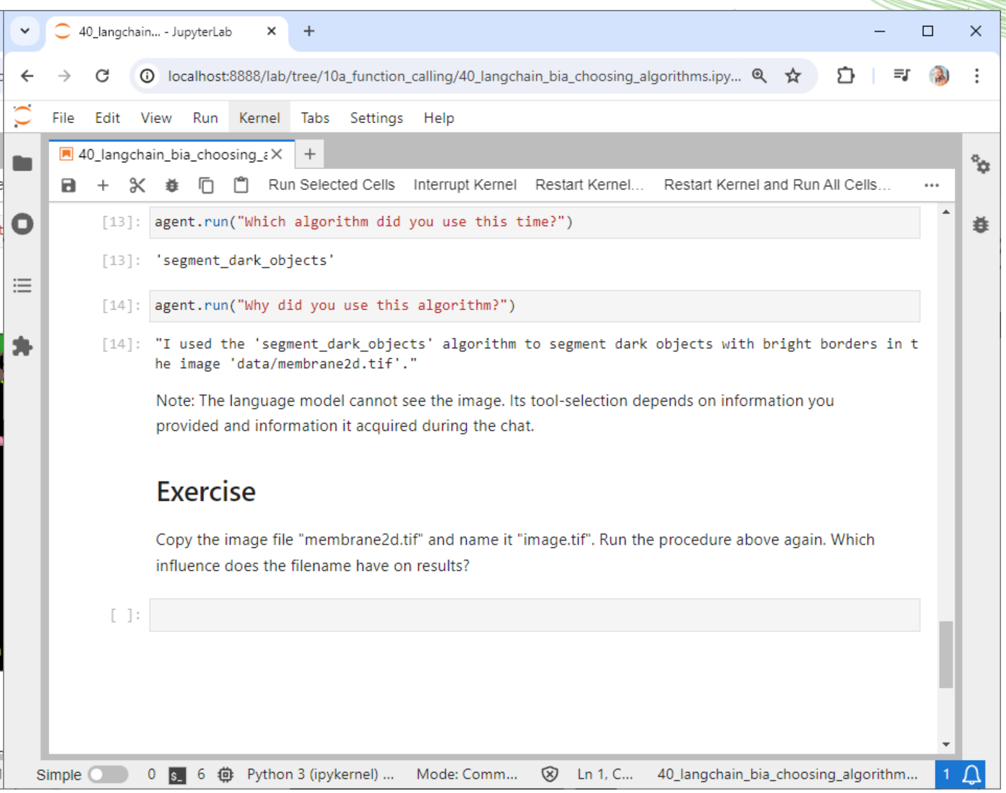
```
[1]: from langchain.memory import ConversationBufferMemory
     from langchain.chat_models import ChatOpenAI
     from langchain.agents import initialize_agent
     from langchain.agents import AgentType
     from langchain.tools import tool

     from skimage.io import imread
     from napari_segment_blobs_and_things_with_membranes import voronoi_otsu_labeling, local_m

     import stackview
```

Again, we define an image storage and a list of tools.

```
[2]: image_storage = {}
     tools = []
```



```
[11]: agent.run("Please show the segmented 'data/membrane2d.t
      showing segmented_data/membrane2d.tif
```



```
[13]: agent.run("Which algorithm did you use this time?")
[13]: 'segment_dark_objects'
[14]: agent.run("Why did you use this algorithm?")
[14]: "I used the 'segment_dark_objects' algorithm to segment dark objects with bright borders in t
      he image 'data/membrane2d.tif'."
```

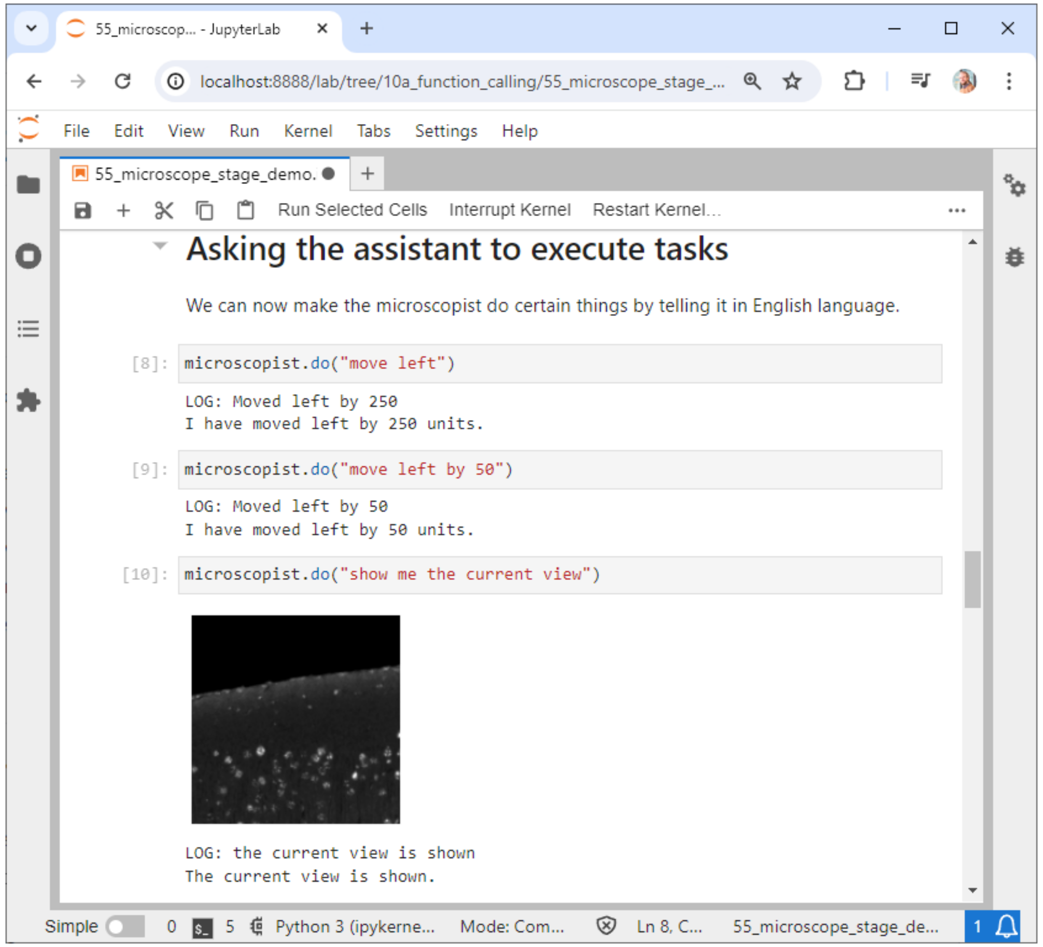
Note: The language model cannot see the image. Its tool-selection depends on information you provided and information it acquired during the chat.

Exercise

Copy the image file "membrane2d.tif" and name it "image.tif". Run the procedure above again. Which influence does the filename have on results?

Exercise: bla-bla-do

- Add zoom-capabilities to the AI-controlled microscope

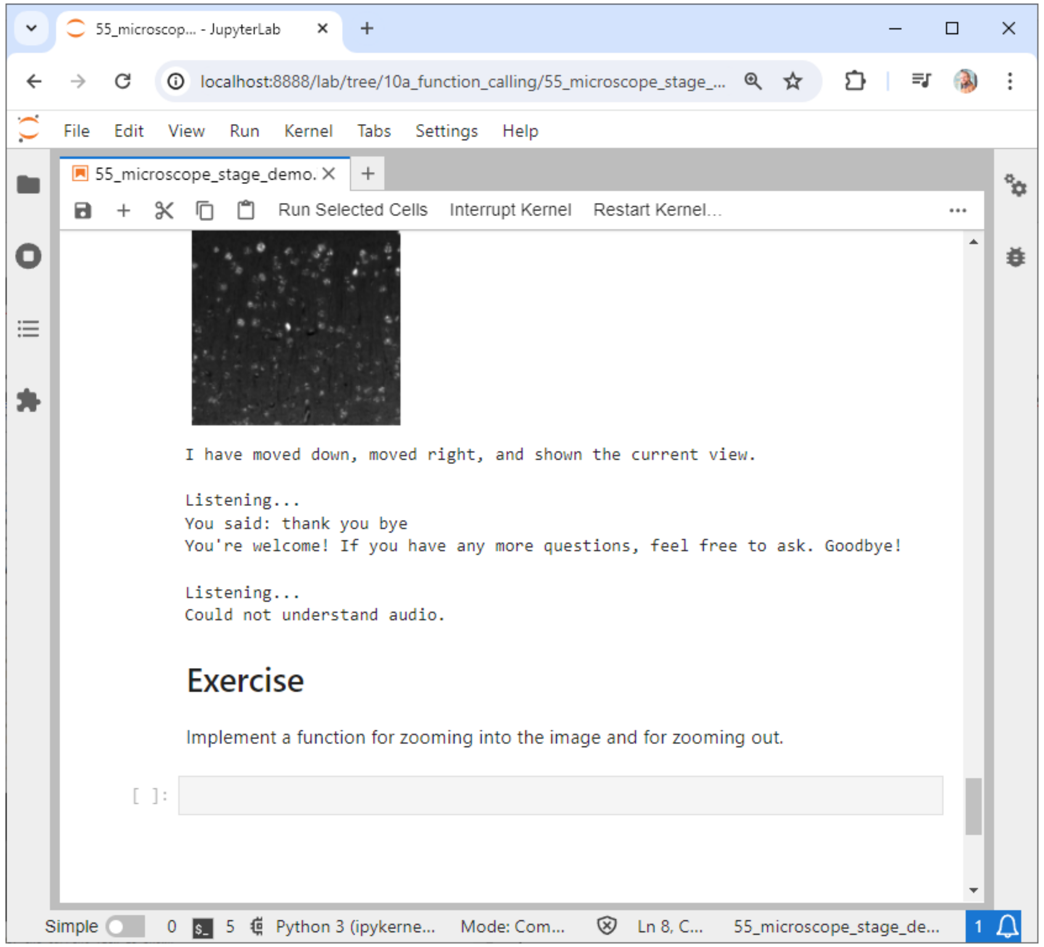

55_microscop... - JupyterLab
localhost:8888/lab/tree/10a_function_calling/55_microscope_stage_...
File Edit View Run Kernel Tabs Settings Help
55_microscope_stage_demo.
Run Selected Cells Interrupt Kernel Restart Kernel...
Simple 0 5 Python 3 (ipykerne... Mode: Com... Ln 8, C... 55_microscope_stage_de...

Asking the assistant to execute tasks

We can now make the microscopist do certain things by telling it in English language.

```
[8]: microscopist.do("move left")

LOG: Moved left by 250
I have moved left by 250 units.

[9]: microscopist.do("move left by 50")

LOG: Moved left by 50
I have moved left by 50 units.

[10]: microscopist.do("show me the current view")
```

```
LOG: the current view is shown
The current view is shown.
```


55_microscop... - JupyterLab
localhost:8888/lab/tree/10a_function_calling/55_microscope_stage_...
File Edit View Run Kernel Tabs Settings Help
55_microscope_stage_demo.
Run Selected Cells Interrupt Kernel Restart Kernel...
Simple 0 5 Python 3 (ipykerne... Mode: Com... Ln 8, C... 55_microscope_stage_de...

```
I have moved down, moved right, and shown the current view.

Listening...
You said: thank you bye
You're welcome! If you have any more questions, feel free to ask. Goodbye!

Listening...
Could not understand audio.
```

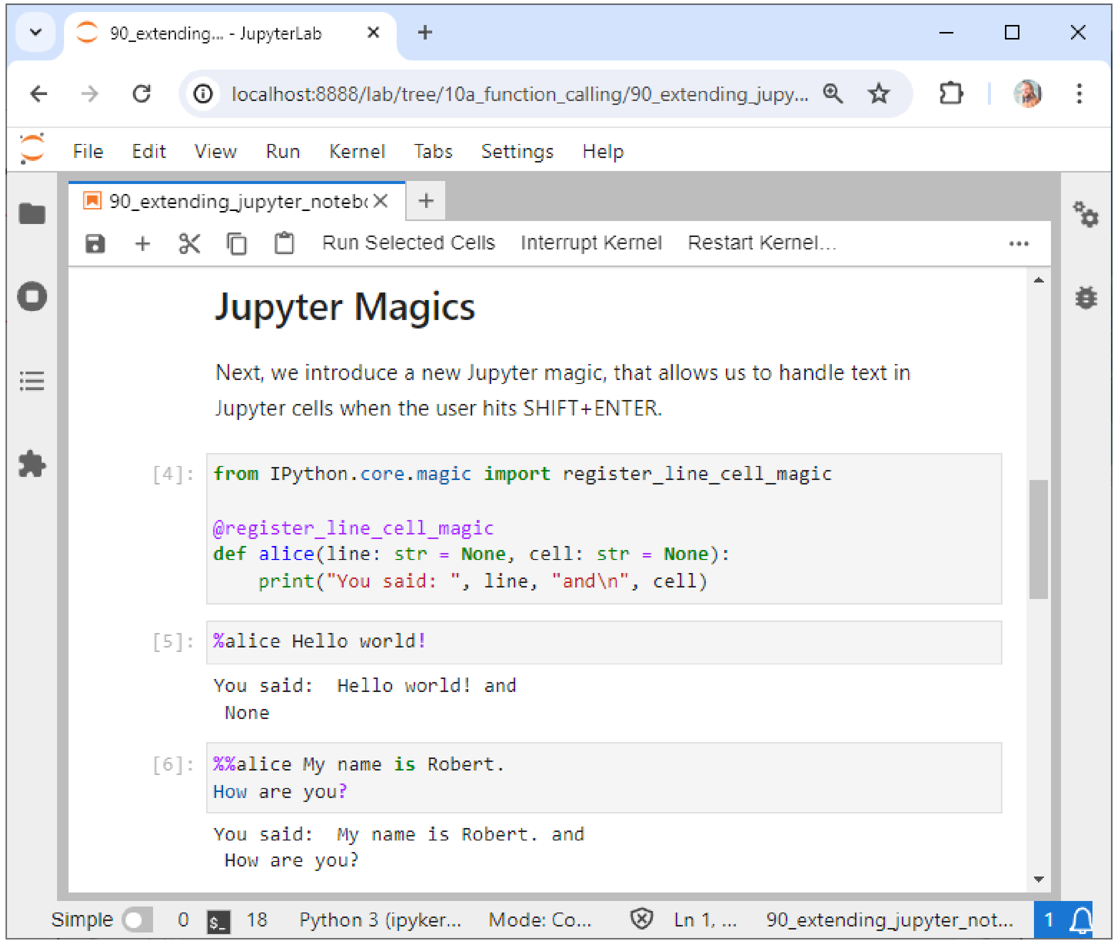
Exercise

Implement a function for zooming into the image and for zooming out.

```
[ ]:
```

Exercise: Jupyter magics

- Build a Jupyter-based chatbot that can process images.



Jupyter Magics

Next, we introduce a new Jupyter magic, that allows us to handle text in Jupyter cells when the user hits SHIFT+ENTER.

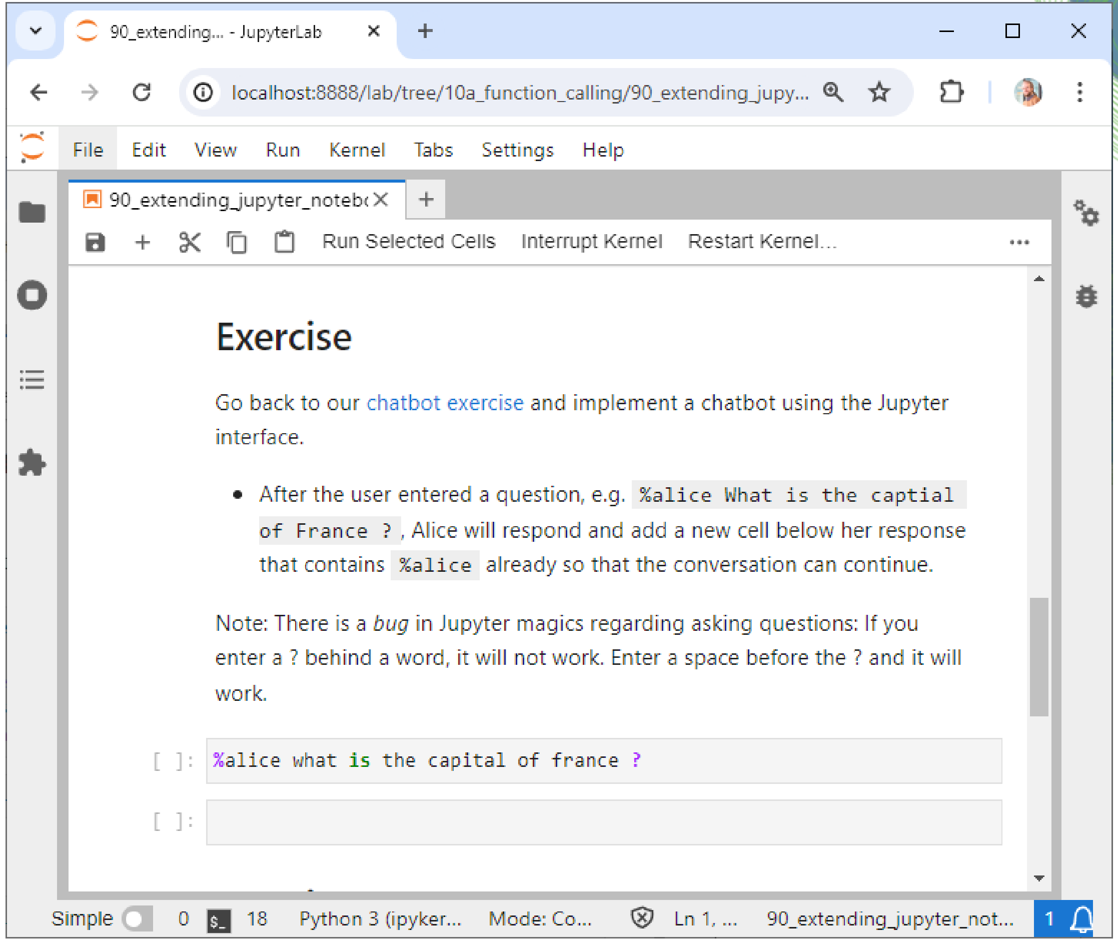
```
[4]: from IPython.core.magic import register_line_cell_magic

     @register_line_cell_magic
     def alice(line: str = None, cell: str = None):
         print("You said: ", line, "and\n", cell)
```

```
[5]: %alice Hello world!
```

```
You said:  Hello world! and
 None
```

```
[6]: %%alice My name is Robert.
     How are you?
```

```
You said:  My name is Robert. and
 How are you?
```



Exercise

Go back to our chatbot exercise and implement a chatbot using the Jupyter interface.

- After the user entered a question, e.g. `%alice What is the captial of France ?`, Alice will respond and add a new cell below her response that contains `%alice` already so that the conversation can continue.

Note: There is a *bug* in Jupyter magics regarding asking questions: If you enter a ? behind a word, it will not work. Enter a space before the ? and it will work.

```
[ ]: %alice what is the capital of france ?
```

```
[ ]:
```