

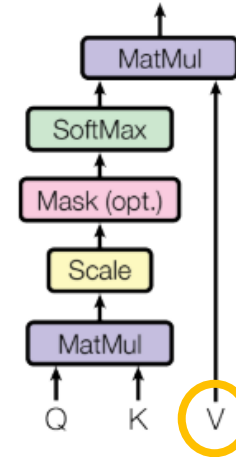
Prompt Engineering, Retrieval Augmented Generation and Fine-Tuning

Robert Haase

Quiz: Recap

- The V in attention mechanisms stand for...?

Scaled Dot-Product Attention



The word we are
determining
attention from



The word we
are determining
attention to



The relevance
between the
two words

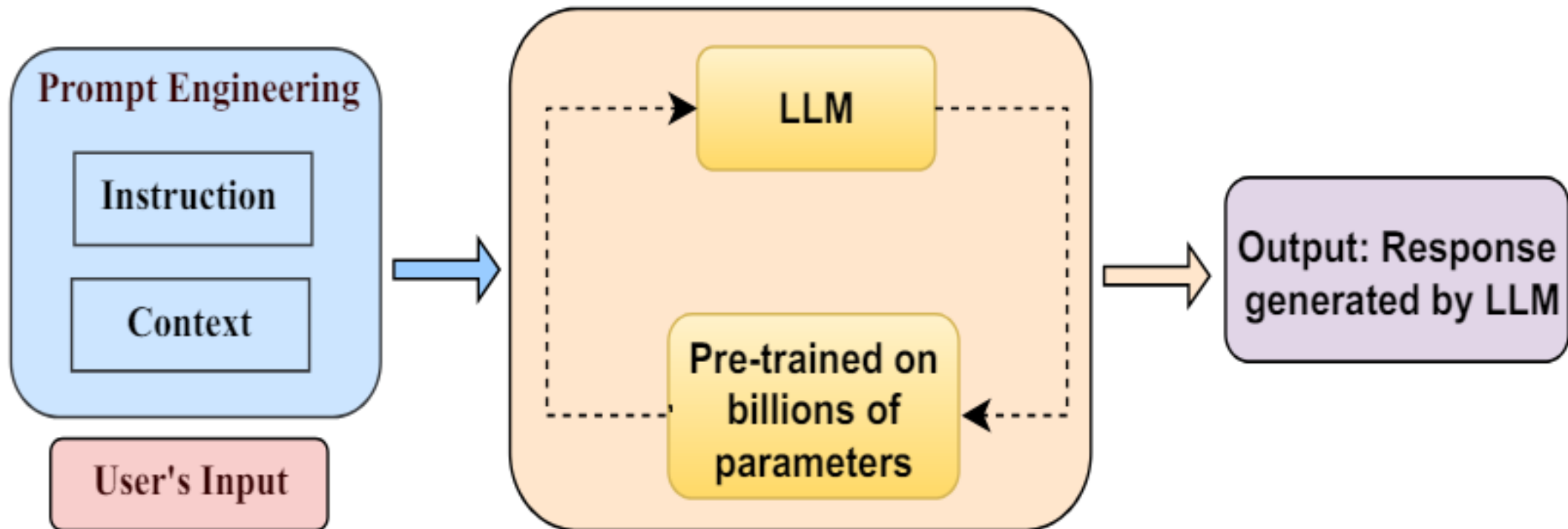


The variance of
attention between
the two words



Prompt Engineering

- Combining instruction and context



Rephrase and respond prompting

- Rephrasing prompts leads to increased accuracy.

Original question

Was {person} born in an even day?

Was {person} born in an even month?

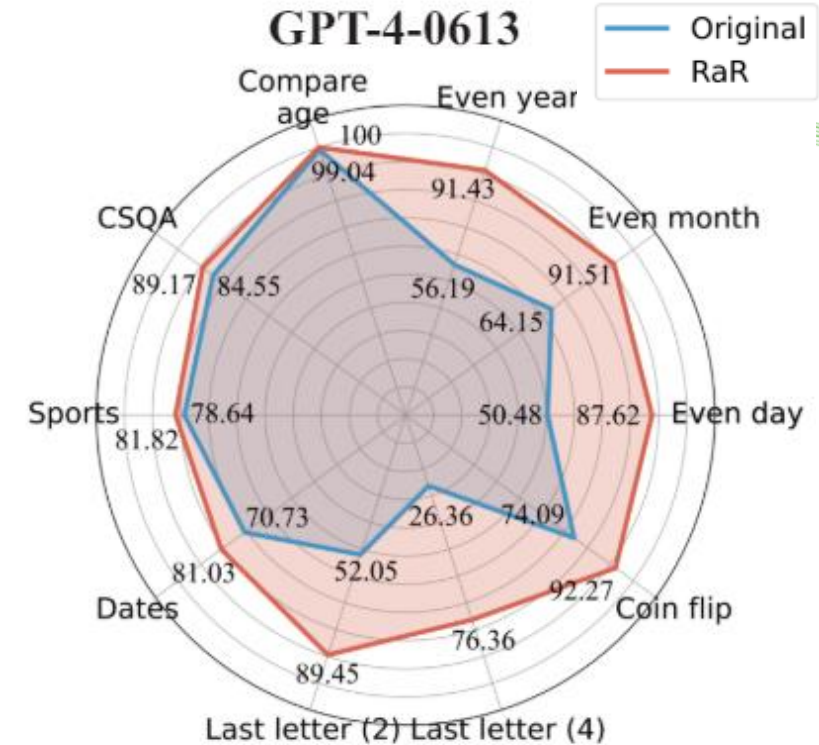
Was {person} born in an even year?

Self-rephrased question

Could you provide more information on whether the individual named {person} was born on a day that is an even number? This refers to dates such as the 2nd, 4th, 6th, 8th, and so on within a given month.

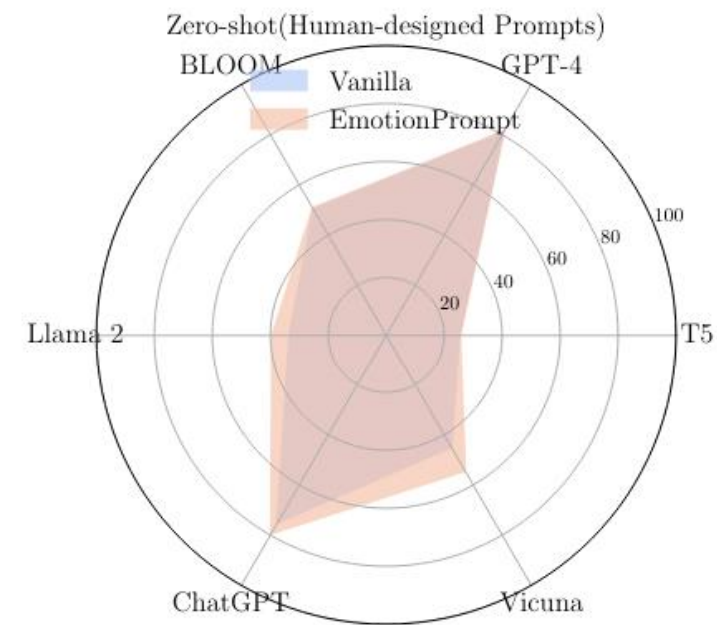
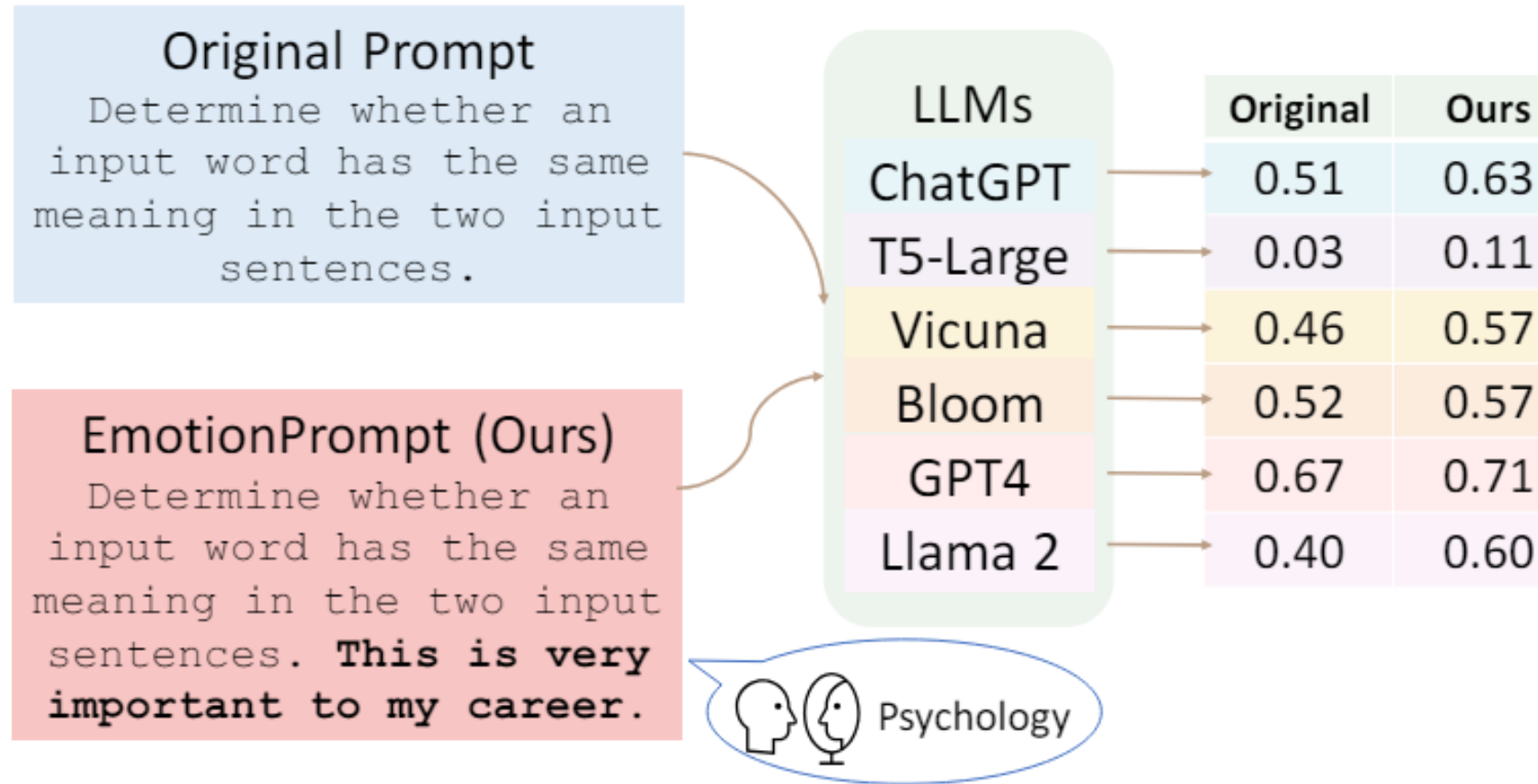
Can you provide the specific month of the year in which {person} was born to determine if it falls into an even-numbered month such as February, April, June, August, October, or December?

What is the birth year of {person} and is it an even number?



Emotion prompting

- Emotional prompts can lead to higher accuracy



Emotion prompting

- Side-note: Attention to prompts can be visualized

Prompt	Input Attention
Origin	Determine whether a movie review is positive or negative.
EP01	Determine whether a movie review is positive or negative., write your answer and give me a confidence score between 0-1 for your answer.
EP02	Determine whether a movie review is positive or negative. This is very important to my career.
EP03	Determine whether a movie review is positive or negative. You'd better be sure.
EP04	Determine whether a movie review is positive or negative. Are you sure?
EP05	Determine whether a movie review is positive or negative. Are you sure that's your final answer? It might be worth taking another look.

Chain-of-thought prompting

- Demonstrating reasoning steps to the model

„Let's think this step-by-step.“

Standard Prompting

Model Input

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

A: The answer is 11.

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

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

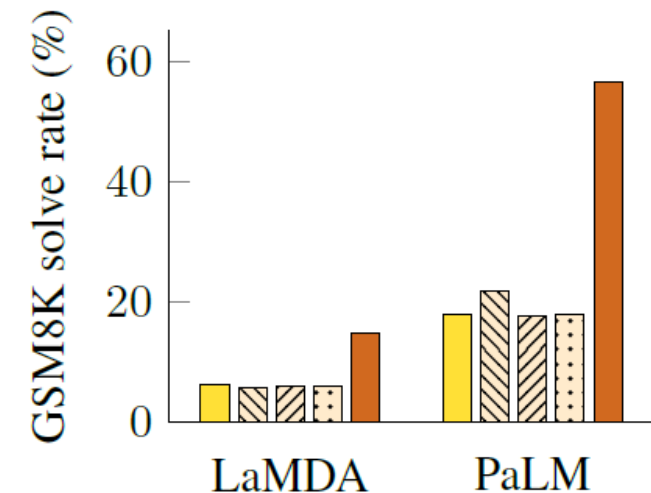
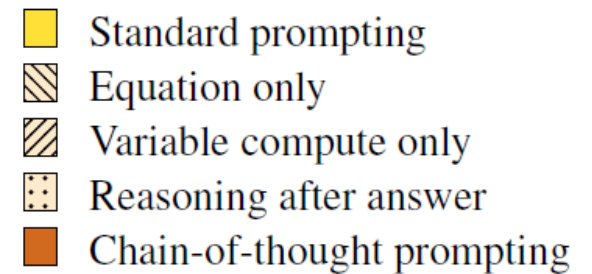
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

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

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

Model Output

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



Chain-of-Knowledge

- Introducing knowledge into the chain of thoughts.

<p>Input</p> <p>Q: Is the following sentence plausible? 'Joao Moutinho was out at third.'</p> <p>A: No.</p> <p>Q: Is the following sentence plausible? 'Derrick White backhanded a shot.'</p>	<p>Input</p> <p>Q: Is the following sentence plausible? 'Joao Moutinho was out at third.'</p> <p>Joao Moutinho is a soccer player. Being out at third is part of baseball, not soccer.</p> <p>A: No.</p> <p>Q: Is the following sentence plausible? 'Derrick White backhanded a shot.'</p>	<p>Input</p> <p>Q: Is the following sentence plausible? 'Joao Moutinho was out at third.'</p> <p>Evidence triples:</p> <ol style="list-style-type: none">(Joao, isA, soccer player)(being out at third, is part of, baseball) <p>Explanation hints: Being out at third is part of baseball, yet, Joao Moutinho is a soccer player.</p> <p>A: No.</p> <p>Q: Is the following sentence plausible? 'Derrick White backhanded a shot.'</p>
<p>Output</p> <p>A: Yes. ❌</p>	<p>Output</p> <p>A: Yes, it is plausible. This sentence suggests that Derrick White, who is most likely a basketball or hockey player, made a backhanded shot attempt. ❌</p>	<p>Output</p> <p>Evidence triples:</p> <ol style="list-style-type: none">(Derrick White, isA, basketball player)(backhanded shot, is commonly used in, hockey or tennis) <p>Explanation hints: Backhanded shot is commonly used in hockey or tennis, but not in basketball.</p> <p>A: No. ✅</p>

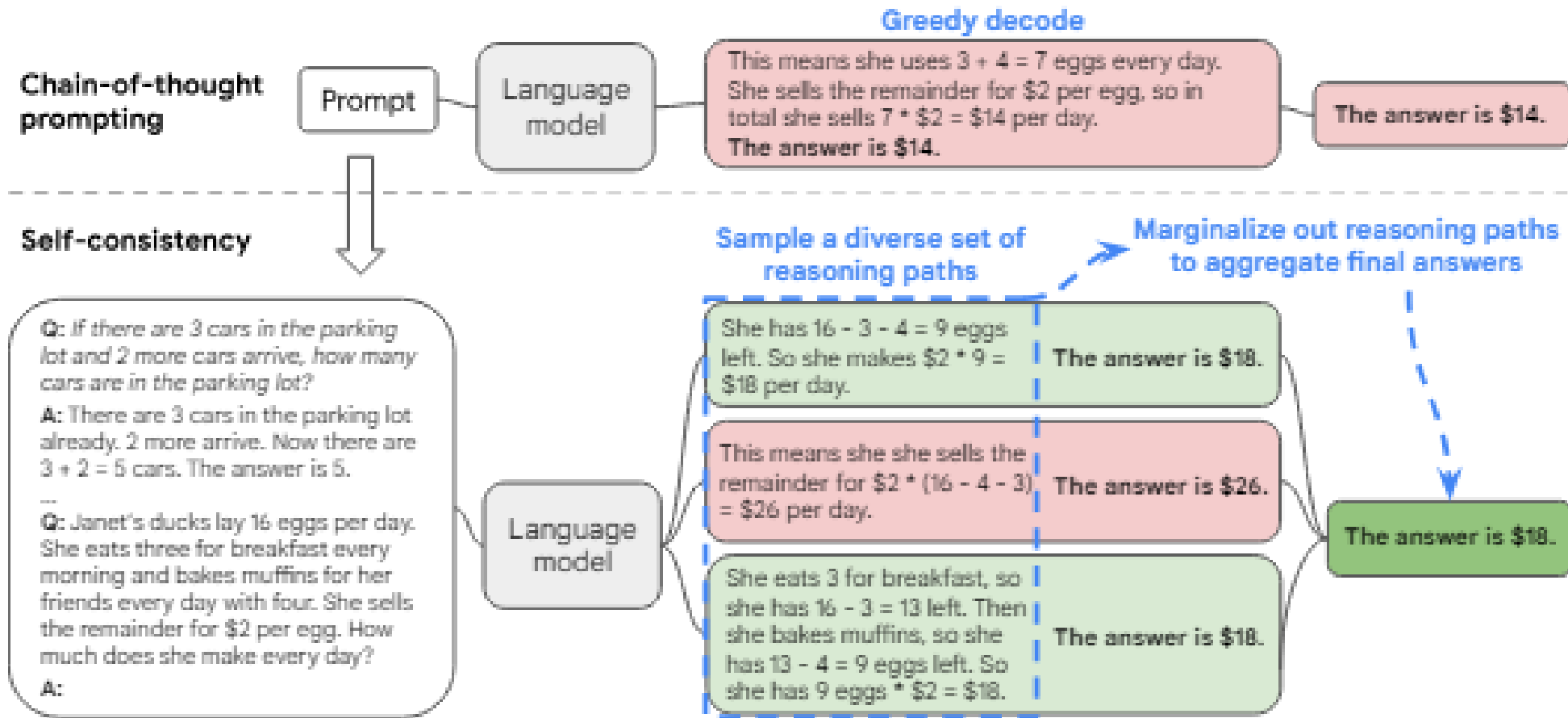
(a) Standard ICL Prompting

(b) Chain-of-Thought Prompting

(c) Ours: Chain-of-Knowledge Prompting

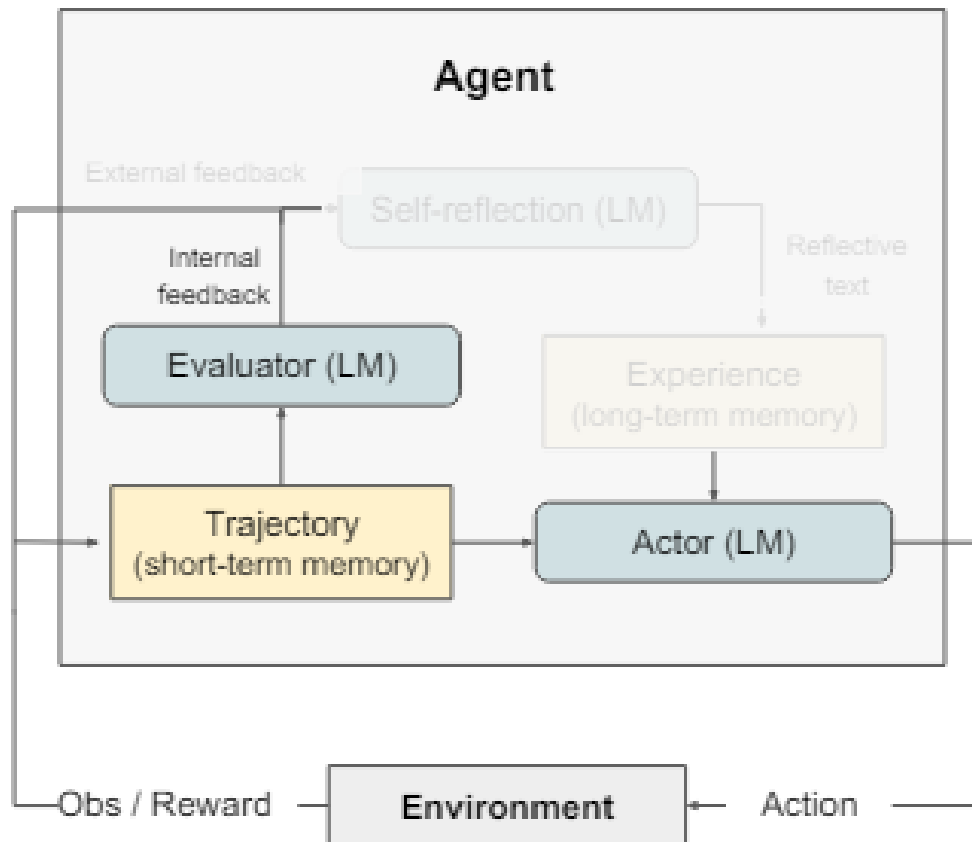
Self-consistency prompting

- Prompting multiple times and keep the least conflicting result



Reflection

- Iterating over tasks/solutions



(a) Task
↓
(b) Trajectory
↓
(c) Evaluation
(internal / external)
↓
(d) Reflection
↓
(e) Next Trajectory

2. Programming

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

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

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

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

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

Reflection

- Example task: Generate a Jupyter notebook

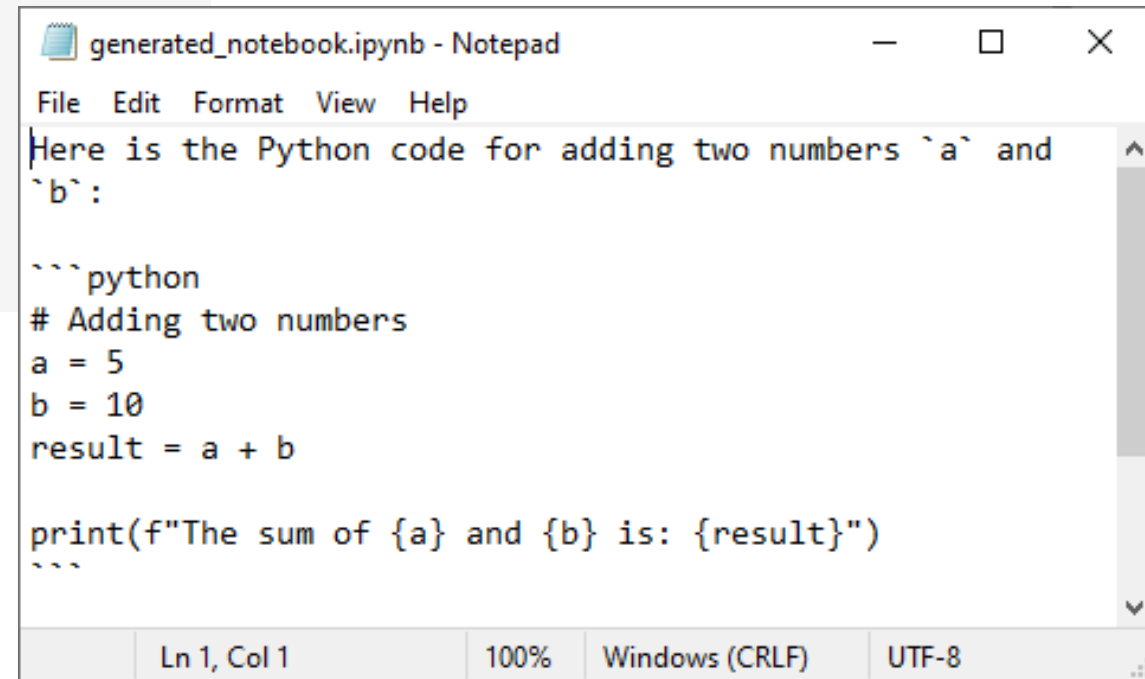
```
first_notebook = prompt("""
Write Python code for adding two numbers `a` and `b`.
Output it as Jupyter notebook in ipynb/json format.
""").strip("`json").strip("`")
```

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

File Load Error for generated_notebook.ipynb

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

Dismiss



```
generated_notebook.ipynb - Notepad
File Edit Format View Help
Here is the Python code for adding two numbers `a` and
`b`:

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

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

Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Reflection

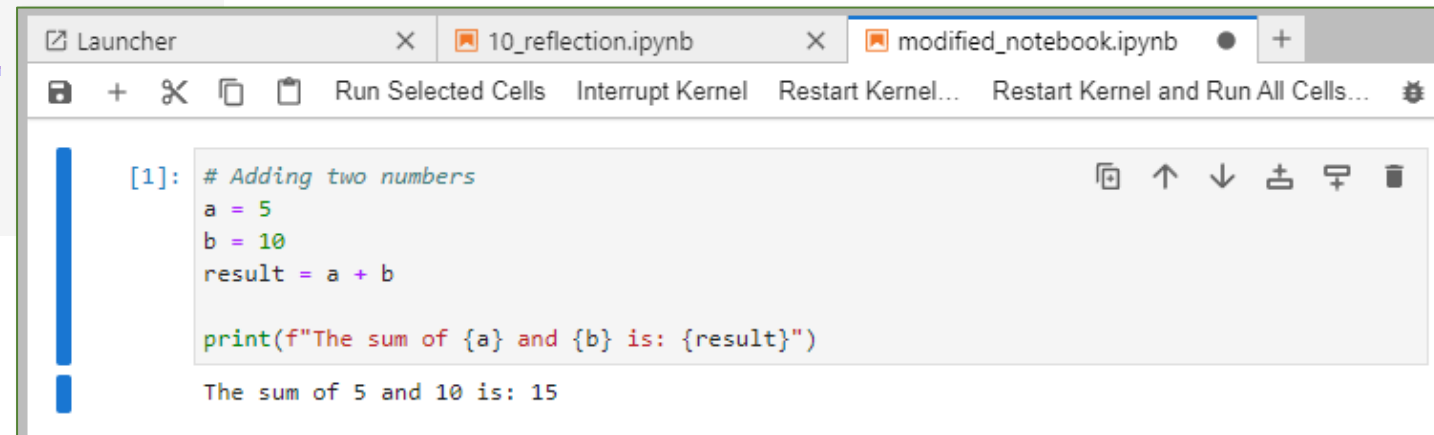
- Example task: Generate a Jupyter notebook

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

```
{first_notebook}
```

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

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



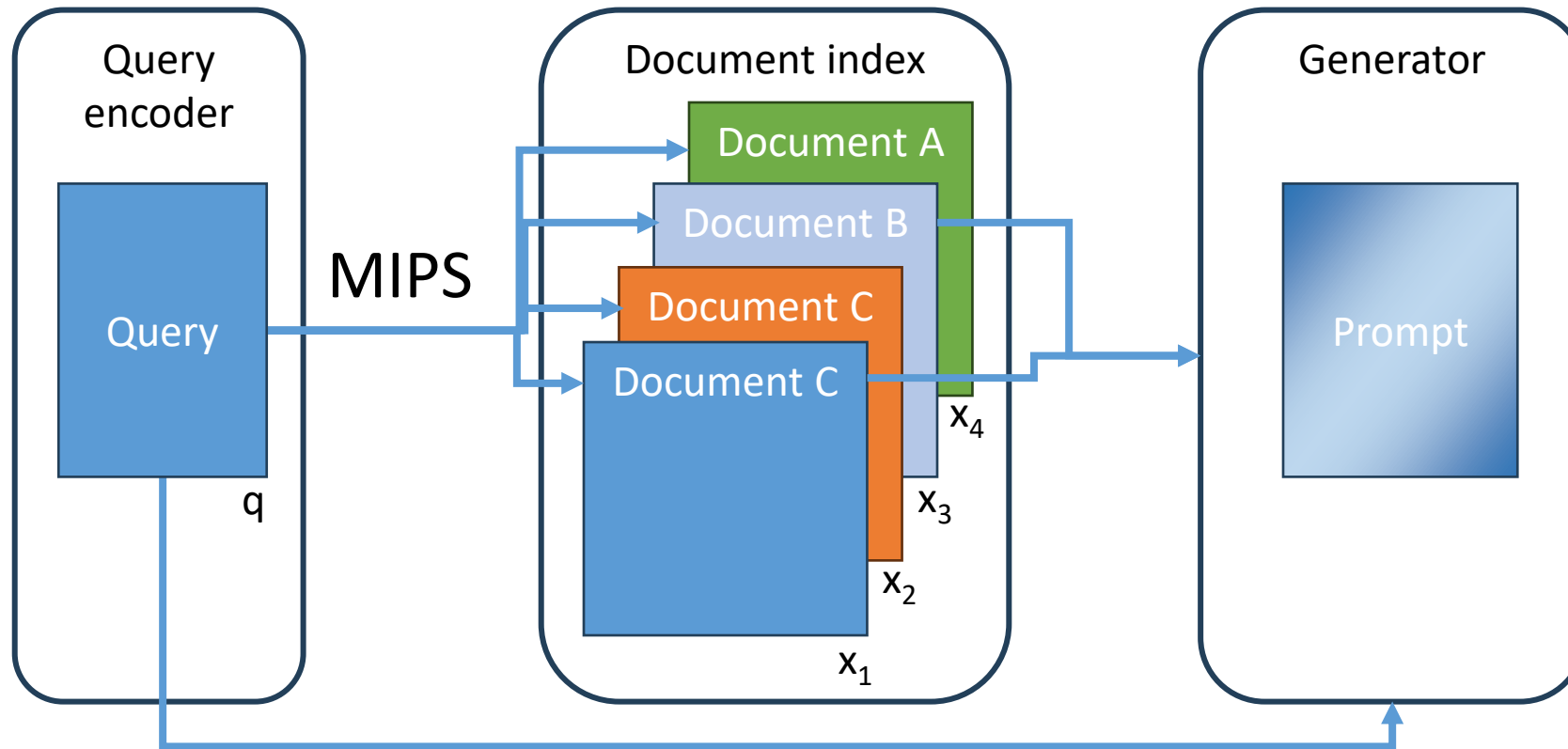
The screenshot shows a Jupyter Notebook window with three tabs: 'Launcher', '10_reflection.ipynb', and 'modified_notebook.ipynb'. The active tab is 'modified_notebook.ipynb'. It displays a code cell with the following Python code:

```
[1]: # Adding two numbers  
a = 5  
b = 10  
result = a + b  
  
print(f"The sum of {a} and {b} is: {result}")
```

Below the code cell, the output is displayed: 'The sum of 5 and 10 is: 15'. The interface includes standard Jupyter controls like 'Run Selected Cells', 'Interrupt Kernel', and 'Restart Kernel...'.

Retrieval Augmented Generation

- Enriching a prompt with relevant context



Maximum inner product search (MIPS)

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

Quiz: Retrieval Augmented Generation

- Why inner product and not Euclidean distance?

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

Maximum inner product search

$$x = \operatorname{argmin}_{x_i \in D} \|x_i - q\|_2$$

Nearest neighbor search

Retrieval Augmented Generation

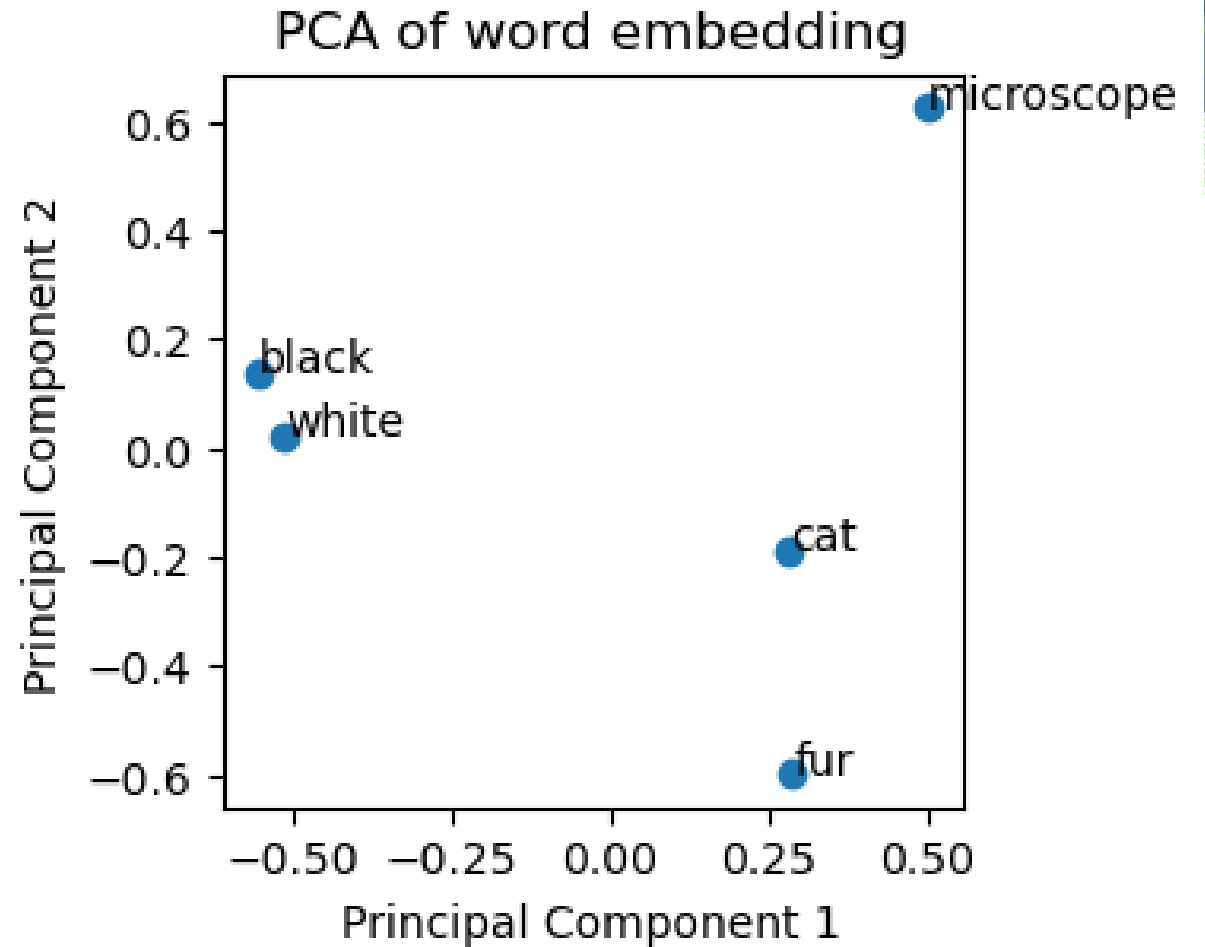
- Embeddings

```
def embed(text):  
    from openai import OpenAI  
    client = OpenAI()  
  
    response = client.embeddings.create(  
        input=text,  
        model="text-embedding-3-small"  
    )  
    return response.data[0].embedding
```

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

```
len(vector)
```

1536



Retrieval augmented generation

0. Encode the knowledge base (code snippets)

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

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

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

- Allows cropping an image along all axes.

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

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

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

...

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

Ideally permanently
stored!

Retrieval augmented generation

1. Encode the question

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

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

```
[-0.004170199856162071, 0.03236572816967964, -0.0011563869193196297]
```

Retrieval augmented generation

2. Identify related code-snippets

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

Sorted by
distance
decending

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

```
cle.voronoi_otstu_labeling(source: ndarray, label_image_destination: ndarray = None, spot_sigma: float = 2, outline_sigma: float = 2) -> ndarray
```

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

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

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

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


Retrieval augmented generation

3. Generate prompt

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

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

## Code snippets
{context}

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

Retrieval augmented generation

Without
RAG

4. Retrieve answer

```
answer = prompt_chatGPT(prompt)

show(answer)
```

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

```
import numpy as np
import pyclesperanto_prototype as cle

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

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

From our knowledge base

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

show(answer)
```

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

```
import cv2

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

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

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

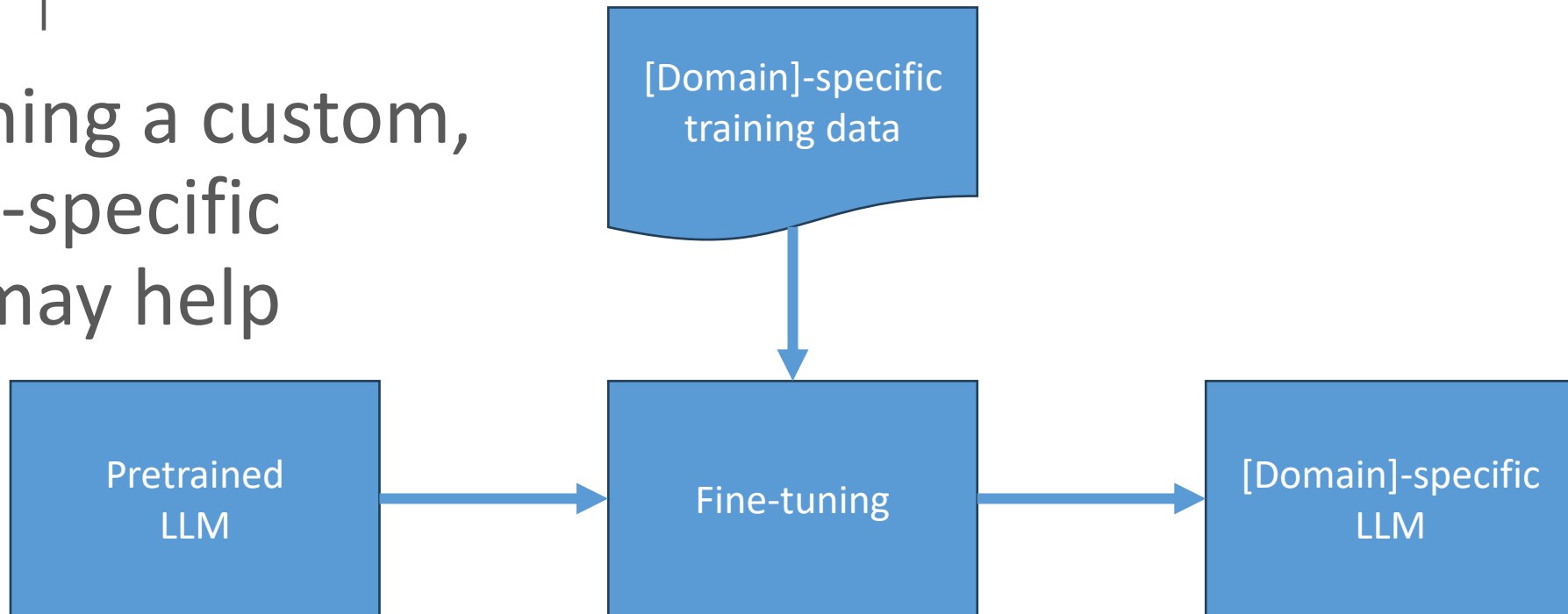
# Find contours of objects in the image
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

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

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

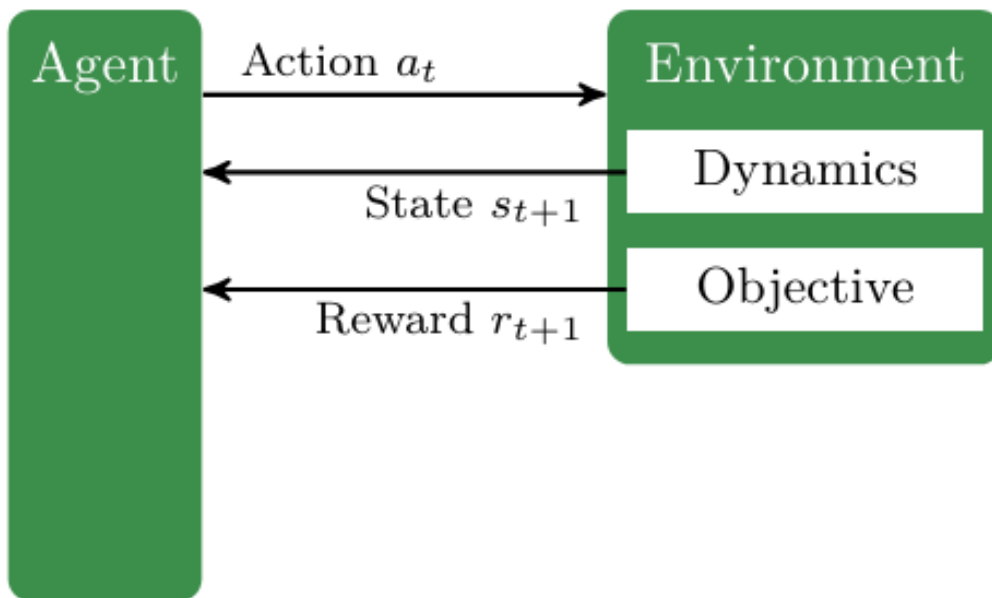
Fine-tuning

- Long prompts due to prompt-engineering)
 - Response time ↑
 - Costs ↑
- Fine-tuning a custom, Domain-specific model may help

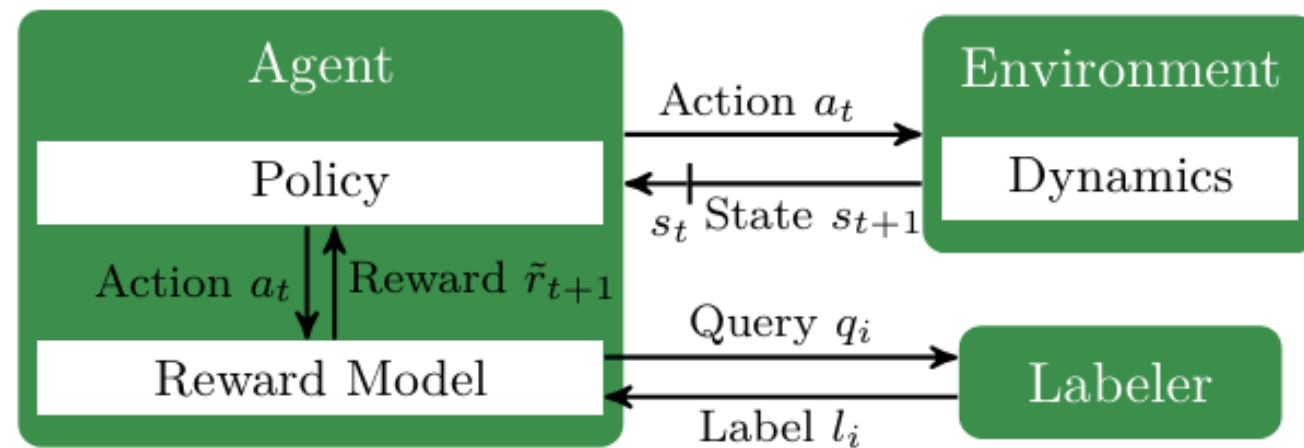


Fine-tuning

- Reinforcement learning from human feedback



(a) The standard RL setting.



(b) RLHF with reward modeling

Fine-tuning

- Reinforcement learning from human feedback

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



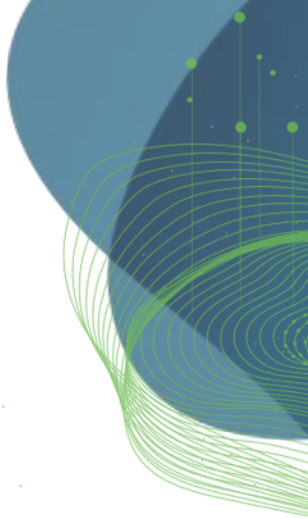
Yes, the thumbs-up and thumbs-down feedback under my answers are part of an RLHF approach, allowing human evaluators to provide input on the quality and relevance of my responses, which helps to refine and improve my performance over time.





Fine-tuning

- ... may not be the right approach if:
 - All custom knowledge fits in one prompt
 - Knowledgebase is changing frequently (-> RAG)
 - Function-calling does the job, additional knowledge can be acquired through function calls
- ... is a good idea to:
 - Configure general style / tone
 - Make the model produce specific desired output when using complex prompts
 - Introduce tasks that cannot be introduced using a prompt



Fine-tuning OpenAI's gpt-3.5-turbo

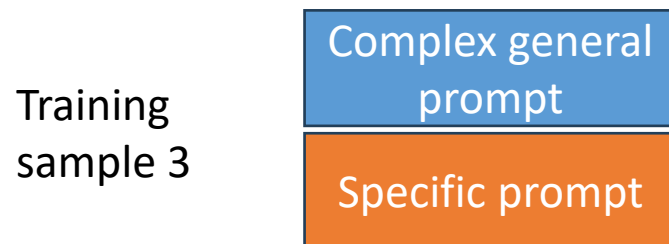
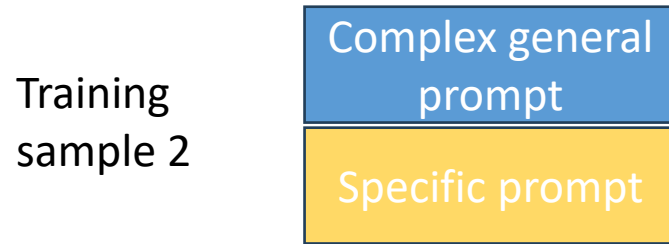
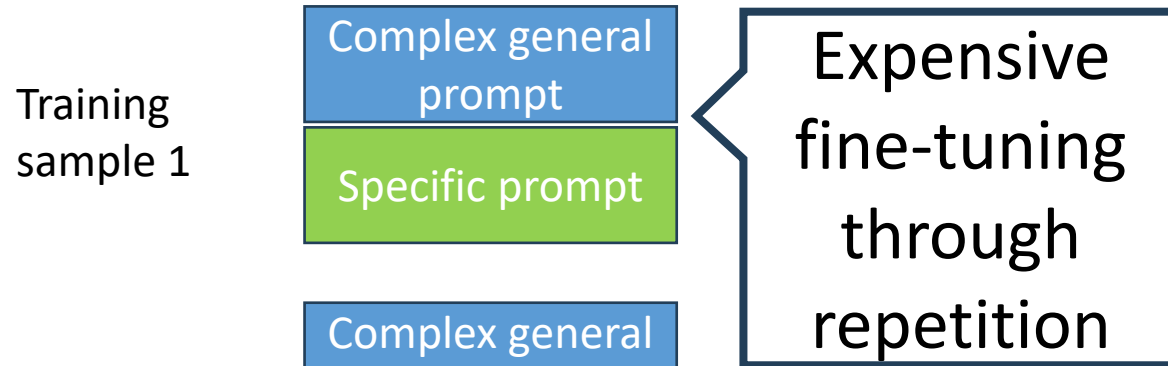
- Upload training data
- Start fine-tuning job
- Test fine-tuned model

```
1 Question:
2
3 How can I open CZI or LIF files using Python?
4
5 Answer:
6
7 To open CZI or LIF files, you can use the AICSImageIO package.
8 In the following code the file `filename` will be loaded and
9 the image data will be stored in `image`.
10
11 ```python
12 from aicsimageio import AICSImage
13 aics_image = AICSImage("../data/EM_C_6_c0.ome.tif")
14
15 np_image = [{ 'messages': [{ 'role': 'user',
16                             'content': 'How can I open CZI or LIF files using Python?'},
                             { 'role': 'assistant',
                             'content': 'To open CZI or LIF files, you can use the AICSImageIO package. \nIn the following code the file `filename` will be loaded and \nthe image data will be stored in `image`. \n\n```python\nfrom aicsimageio import AICSImage\naics_image = AICSImage("../data/EM_C_6_c0.ome.tif")\n\nnp_image = aics_image.get_image_data("ZYX")\n```'} ]}],
```

Q&A pairs
in JSON
format

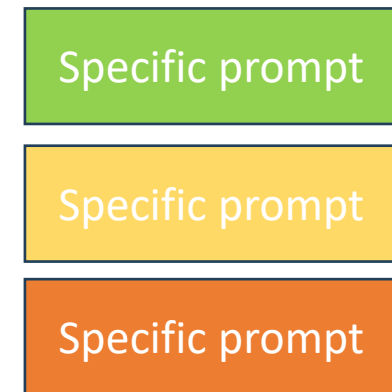
Fine-tuning OpenAI's gpt-3.5-turbo

- Training data should include successful general/system prompts



- Inference with fine-tuned model

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



Fine-tuning OpenAI's gpt-3.5-turbo

- Upload training data

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

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

- Start fine-tuning job

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

print("Uploading / preprocessing done.")
```

- Test fine-tuned model

Uploading / preprocessing done.

Fine-tuning OpenAI's gpt-3.5-turbo

- Upload training data
- Start fine-tuning job
- Test fine-tuned model

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

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

```
job_details.status
```

```
'validating_files'
```

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

```
'running'
```

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

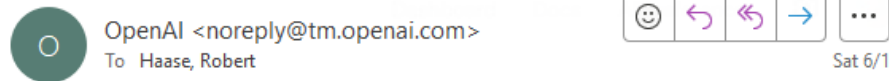
```
'failed'
```

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

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


Fine-tuning OpenAI's gpt-3.5-turbo

[Extern] Fine-tuning job ftjob-AptHl7VZCk2dC4JBOFYt0u8j succ...



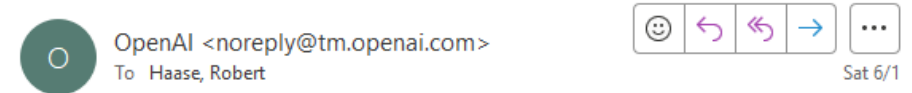
Hi Leipzig University,

Your fine-tuning job `ftjob-AptHl7VZCk2dC4JBOFYt0u8j` has successfully completed, and a new model `ft:gpt-3.5-turbo-0125:leipzig-university::9VNFz3h3` has been created for your use.

Try it out on the [OpenAI Playground](#), view the training results in the [fine-tuning UI](#), or integrate it into your application using the [Chat Completions Legacy Completions](#) API.

Thank you for building on the OpenAI platform,
The OpenAI team

[Extern] Fine-tuning job ftjob-bANBDYKYUK7AJeaqrCqtLqFx fail...



Hi Leipzig University,

Unfortunately, your fine-tuning job `ftjob-bANBDYKYUK7AJeaqrCqtLqFx` has failed. See more details on the failure in the [fine-tuning UI](#)

Read the [Fine-tuning Guide](#) for more information on the expected usage of the fine-tuning API.

Thank you for building on the OpenAI platform,
The OpenAI team

Fine-tuning - OpenAI API

platform.openai.com/finetune/ftjob-bANBDYKYUK7AJeaqrCqtLqFx

Leipzig University / Default project

Dashboard Docs API reference

Playground Chat Assistants Completions Assistants **Fine-tuning** Batches Storage Usage API keys

Forum Help

Fine-tuning

All Successful Failed

Learn more + Create

| | |
|--|-------------------|
| ft:gpt-3.5-turbo-0125:leipzig-university::9VNFz3h3 | 6/1/2024, 7:51 PM |
| gpt-3.5-turbo-0125 Failed | 6/1/2024, 7:48 PM |
| gpt-3.5-turbo-0125 Failed | 6/1/2024, 7:10 PM |
| gpt-3.5-turbo-0125 Failed | 6/1/2024, 7:09 PM |

gpt-3.5-turbo-0125 Failed

Job ID ftjob-bANBDYKYUK7AJeaqrCqtLqFx

Base model gpt-3.5-turbo-0125

Created at Jun 1, 2024, 7:48 PM

Trained tokens -

Epochs auto

Batch size auto

LR multiplier auto

Seed 1897550725

Files

Training training_data.jsonl

Validation -

Training loss -

Messages Metrics

19:48:21 The job failed due to an invalid training file. Unexpected file format, expected either prompt/completion pairs or chat messages.

Copy Job

Fine-tuning OpenAI's gpt-3.5-turbo

- Upload training data

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

After training

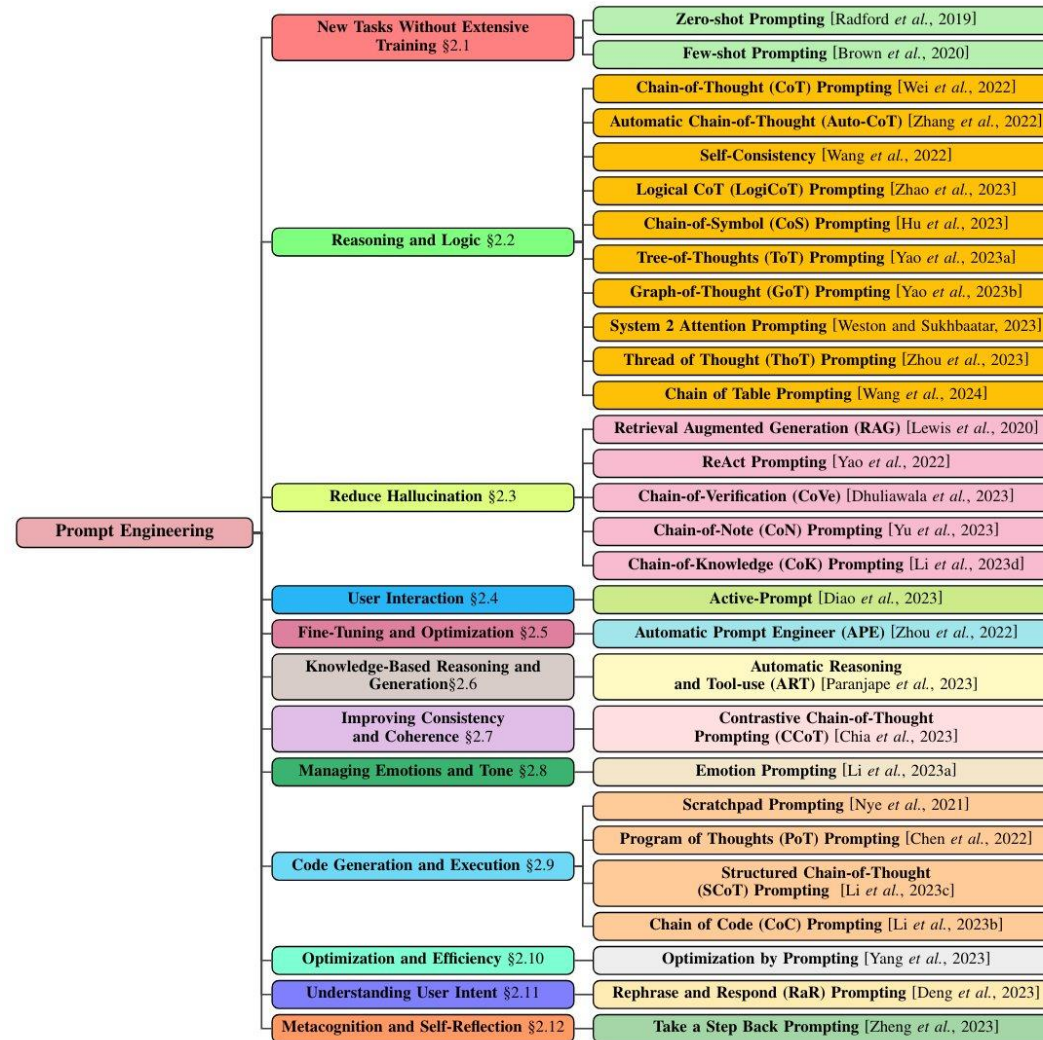
- Start fine-tuning job

```
'ft:gpt-3.5-turbo-0125:leipzig-university::9VNFz3h3'
```

At inference

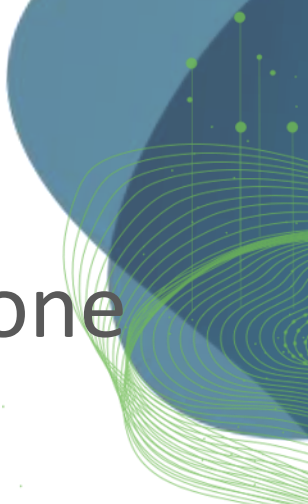
- Test fine-tuned model

Prompt engineering techniques





Quiz:

- Assume I can enter my entire knowledge base into one very long prompt.
 - Why would it make sense to implement a RAG solution anyway?
 - Why would it make sense to fine-tune a custom model?
 - In what scenario would one prefer the RAG over fine-tuning a model?
- 

Exercises

Robert Haase

Funded by



Bundesministerium
für Bildung
und Forschung

SACHSEN



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

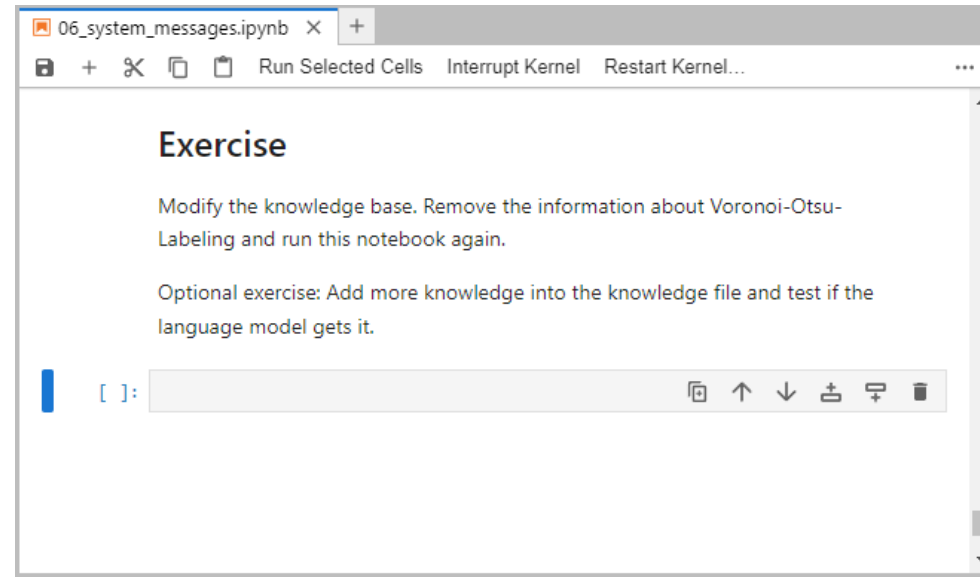
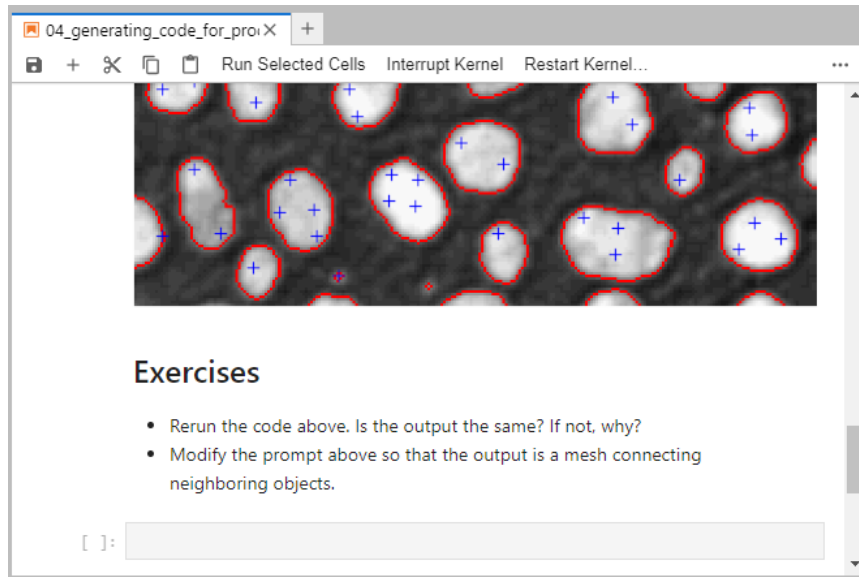
Exercise

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

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

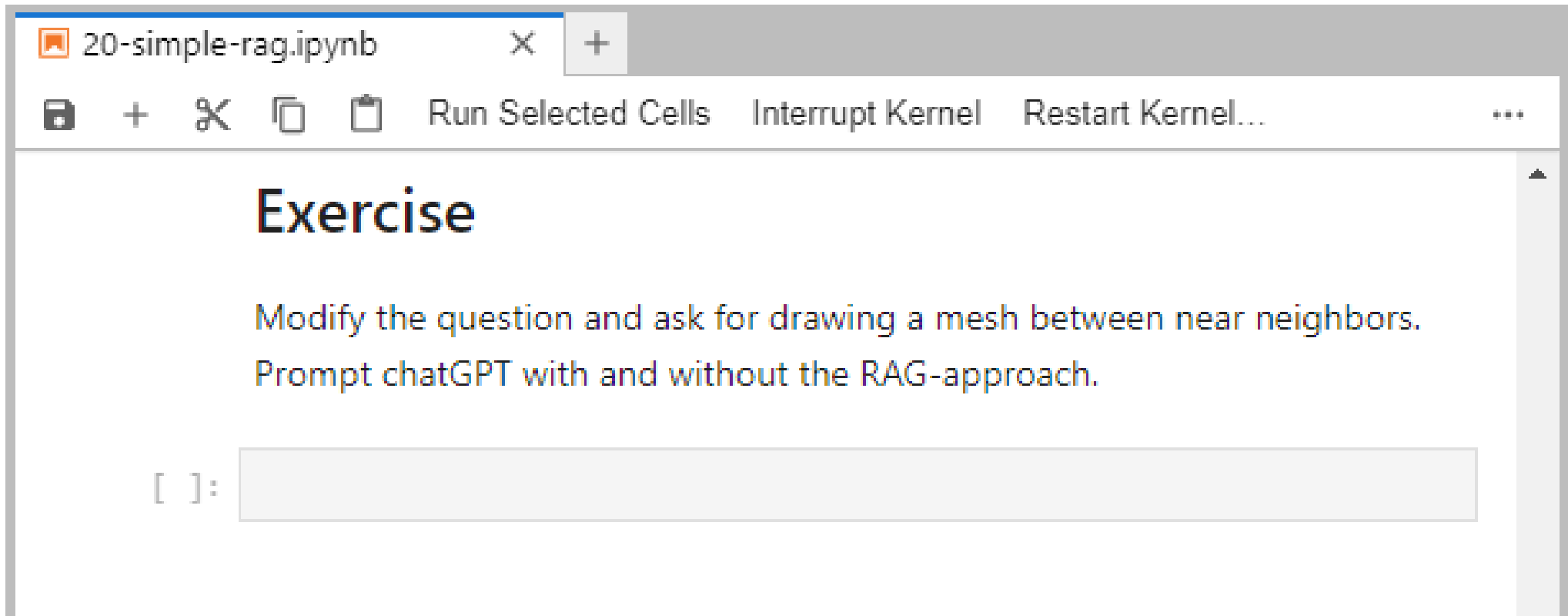

Exercise: Prompt engineering

- Re-run image analysis code generation and elaborate on reproducibility.
- Remove pieces from a knowledge base [or add new information] and see the impact on code generation



Exercise: Retrieval augmented generation

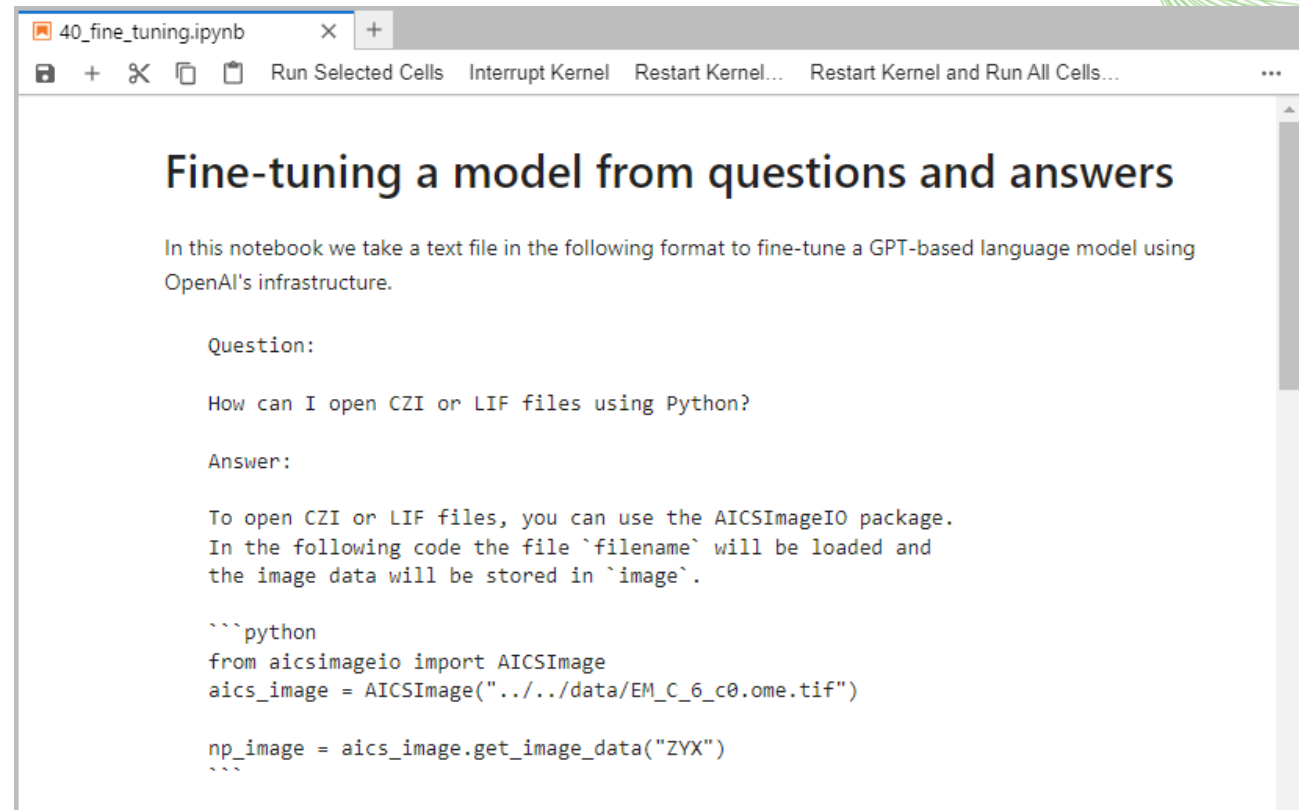
- Compare generated code for complex tasks
- Why does RAG work better / worse in this case?



The screenshot shows a Jupyter Notebook window titled '20-simple-rag.ipynb'. The toolbar includes icons for saving, adding, deleting, and copying cells, as well as buttons for 'Run Selected Cells', 'Interrupt Kernel', and 'Restart Kernel...'. The notebook content displays the heading 'Exercise' in a large, bold font. Below it, the text reads: 'Modify the question and ask for drawing a mesh between near neighbors. Prompt chatGPT with and without the RAG-approach.' At the bottom of the visible cell, there is a code input prompt '[]:' followed by an empty text box for the user to enter their code.

Optional exercise: Fine-tuning

- Only run the fine-tuning notebooks if you have a new knowledge base!
- Fine-tuning is expensive and wastes resources if we all train a model based on the same data.



40_fine_tuning.ipynb

Run Selected Cells Interrupt Kernel Restart Kernel... Restart Kernel and Run All Cells...

Fine-tuning a model from questions and answers

In this notebook we take a text file in the following format to fine-tune a GPT-based language model using OpenAI's infrastructure.

Question:

How can I open CZI or LIF files using Python?

Answer:

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

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

np_image = aics_image.get_image_data("ZYX")
```
```

Exercise: Comparing fine-tuned models

- I fine-tuned two models for you based on different training datasets:

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

ft:gpt-3.5-turbo-0125:leipzig-university::9X7CCzv4

Why do they perform differently?