# Neural Networks

## Large Language Models

Guest lecture by Xufeng Zhao and Dr. Jae Hee Lee

Presenter: Dr. Jae Hee Lee

KNOWLEDGE
TECHNOLOGY

http://www.informatik.uni-hamburg.de/WTM/

# Outline

- How to Train an LLM?
- Unleashing The Power of an LLM
- How to Interpret an LLM?
- LLM Applications in Robotics

# A Short History of Language Models

A Survey of Large Language Models
Image Source

# A Short History of Language Models

- 1950-1980
  - Rule-based approaches

A Survey of Large Language Models
Image Source

# A Short History of Language Models

- 1950-1980
  - Rule-based approaches
- 1980-2000
  - Statistical language models

A Survey of Large Language Models
Image Source

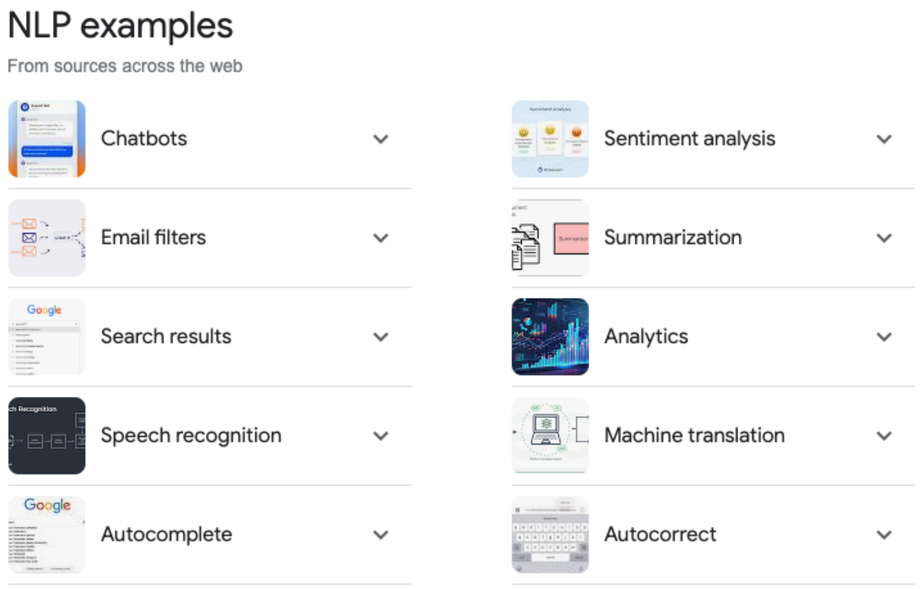
# A Short History of Language Models

- 1950-1980
  - Rule-based approaches
- 1980-2000
  - Statistical language models
- 2000-2010
  - Neural language models

A Survey of Large Language Models
Image Source

# A Short History of Language Models

- 1950-1980
  - Rule-based approaches
- 1980-2000
  - Statistical language models
- 2000-2010
  - Neural language models
- 2010-2020
  - Pre-trained language models



NLP examples

From sources across the web

Chatbots

Sentiment analysis

Email filters

Summarization

Search results

Analytics

Speech recognition

Machine translation

Autocomplete

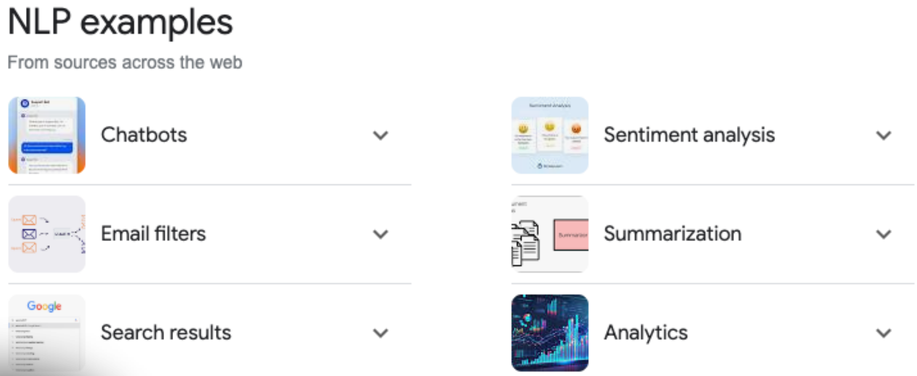
Autocorrect

[A Survey of Large Language Models](#)

[Image Source](#)

# A Short History of Language Models

- 1950-1980
  - Rule-based approaches
- 1980-2000
  - Statistical language models
- 2000-2010
  - Neural language models
- 2010-2020
  - Pre-trained language models
- 2020-Present
  - Large language models
  - (e.g. GPT-4)



NLP examples

From sources across the web

Chatbots

Email filters

Search results

Speech recognition

Sentiment analysis

Summarization

Analytics

Machine translation

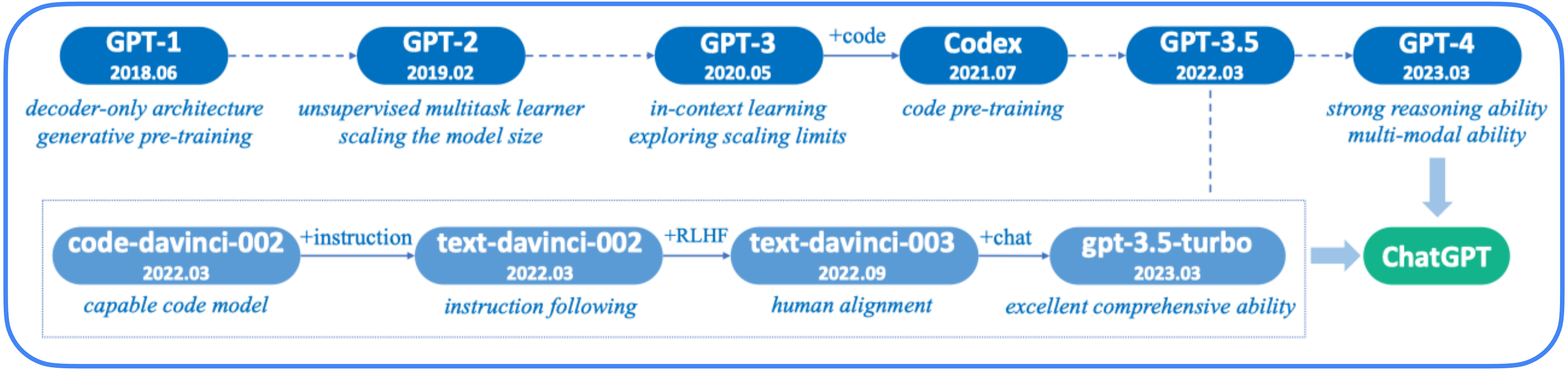
Autocorrect

A Survey of Large Language Models

Image Source

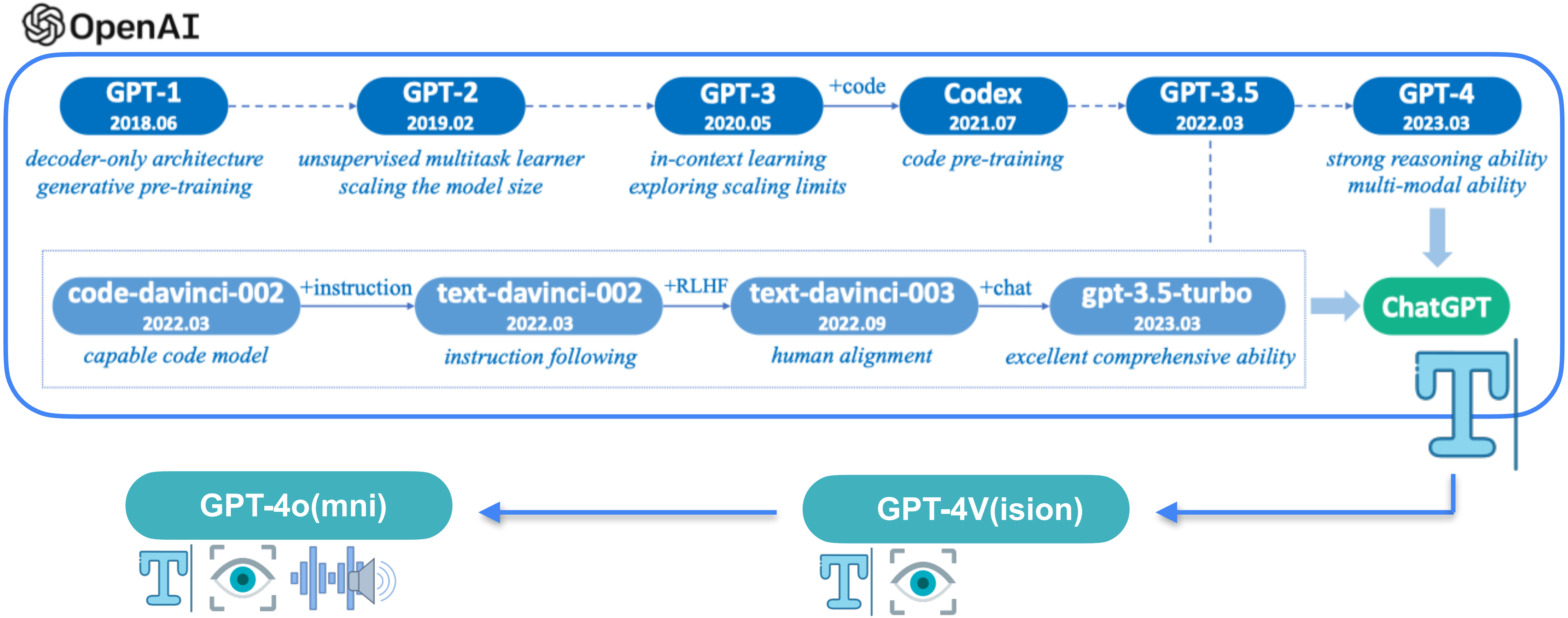
# GPT Development

OpenAI



A Survey of Large Language Models

# GPT Development



OpenAI

**GPT-1** 2018.06 — *decoder-only architecture* *generative pre-training*

**GPT-2** 2019.02 — *unsupervised multitask learner* *scaling the model size*

**GPT-3** 2020.05 — *in-context learning* *exploring scaling limits*

+code

**Codex** 2021.07 — *code pre-training*

**GPT-3.5** 2022.03

**GPT-4** 2023.03 — *strong reasoning ability* *multi-modal ability*

**code-davinci-002** 2022.03 — *capable code model*

+instruction

**text-davinci-002** 2022.03 — *instruction following*

+RLHF

**text-davinci-003** 2022.09 — *human alignment*

+chat

**gpt-3.5-turbo** 2023.03 — *excellent comprehensive ability*

**ChatGPT**

**GPT-4V(ision)**

**GPT-4o(mni)**

A Survey of Large Language Models

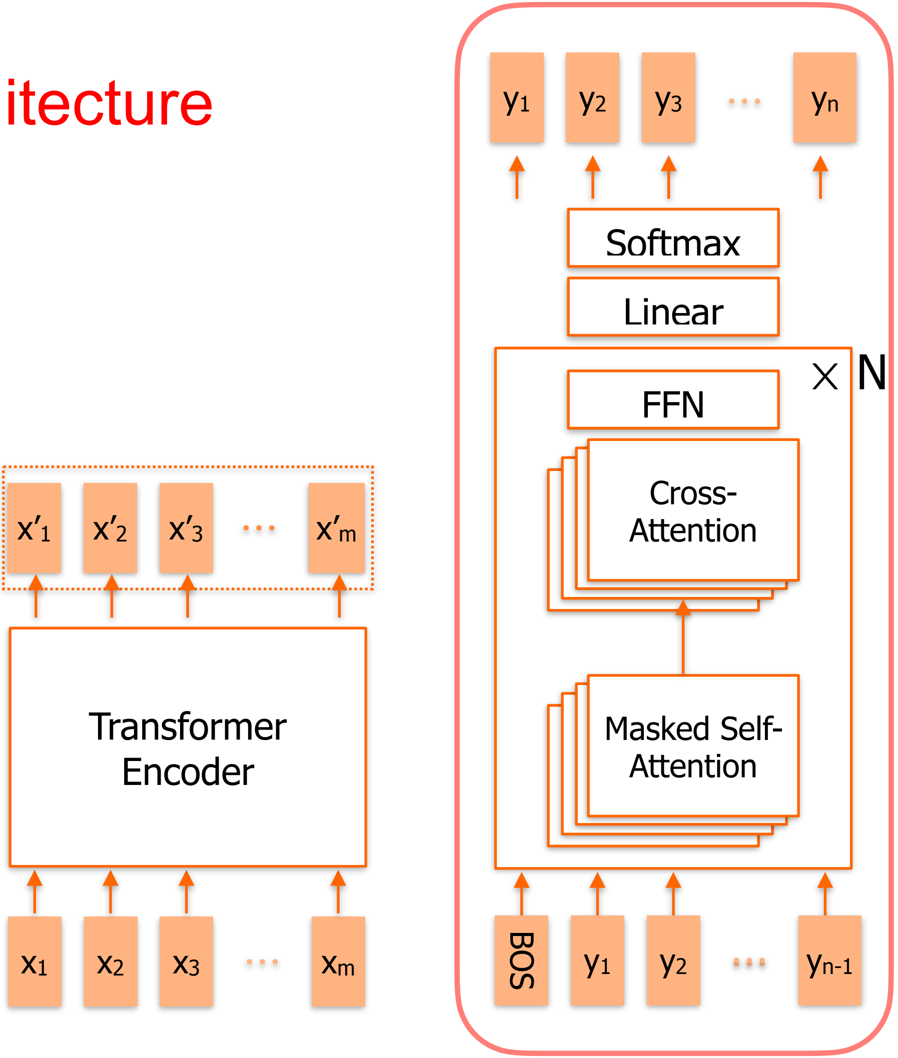
# LLM Architecture

# LLM Architecture

- What is an LLM?

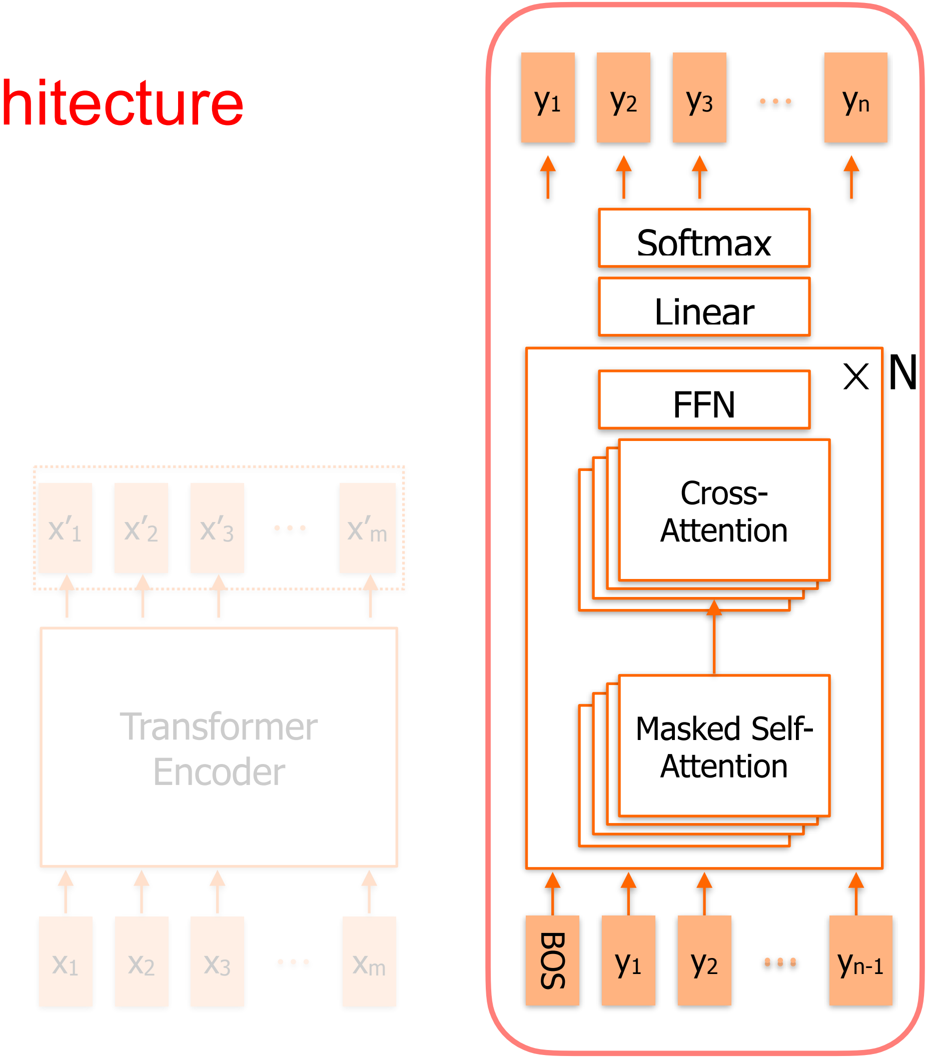
# LLM Architecture

- What is an LLM?
  - An LLM is a **transformer**



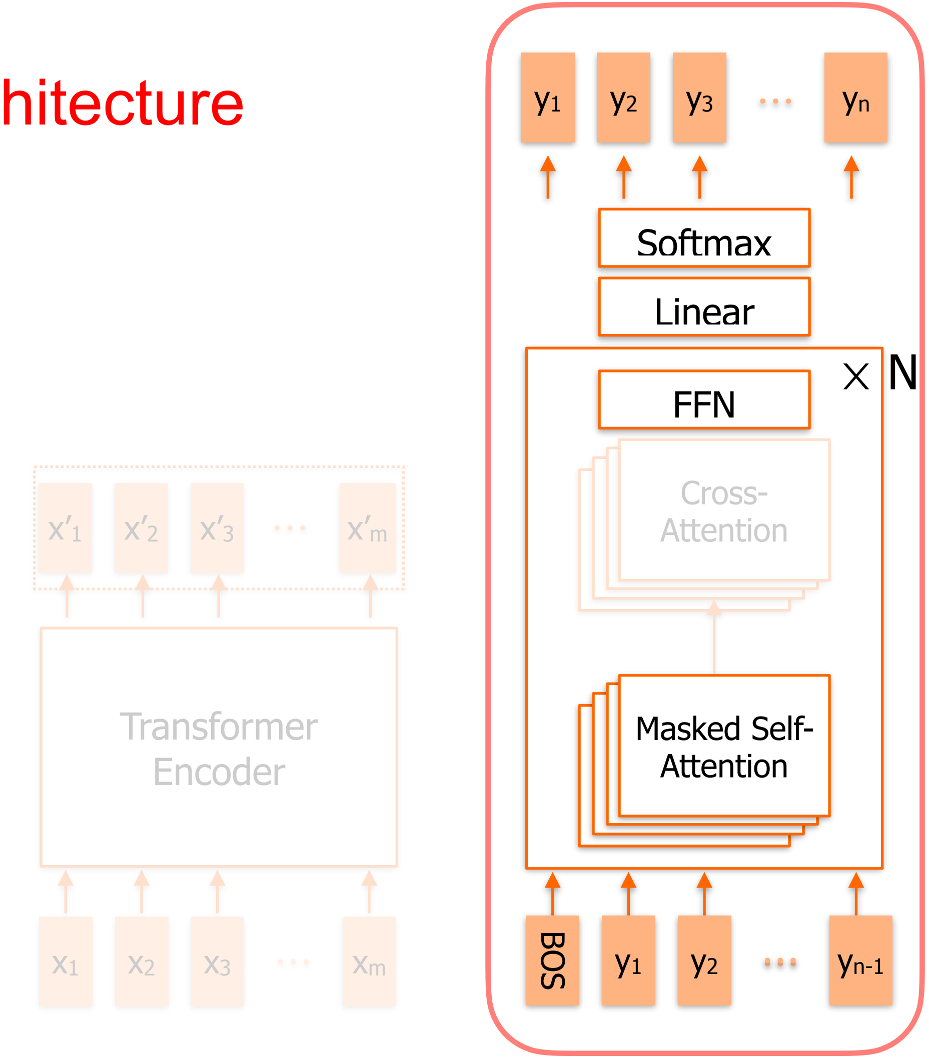
# LLM Architecture

- What is an LLM?
  - An LLM is a **transformer**
  - **decoder-only**



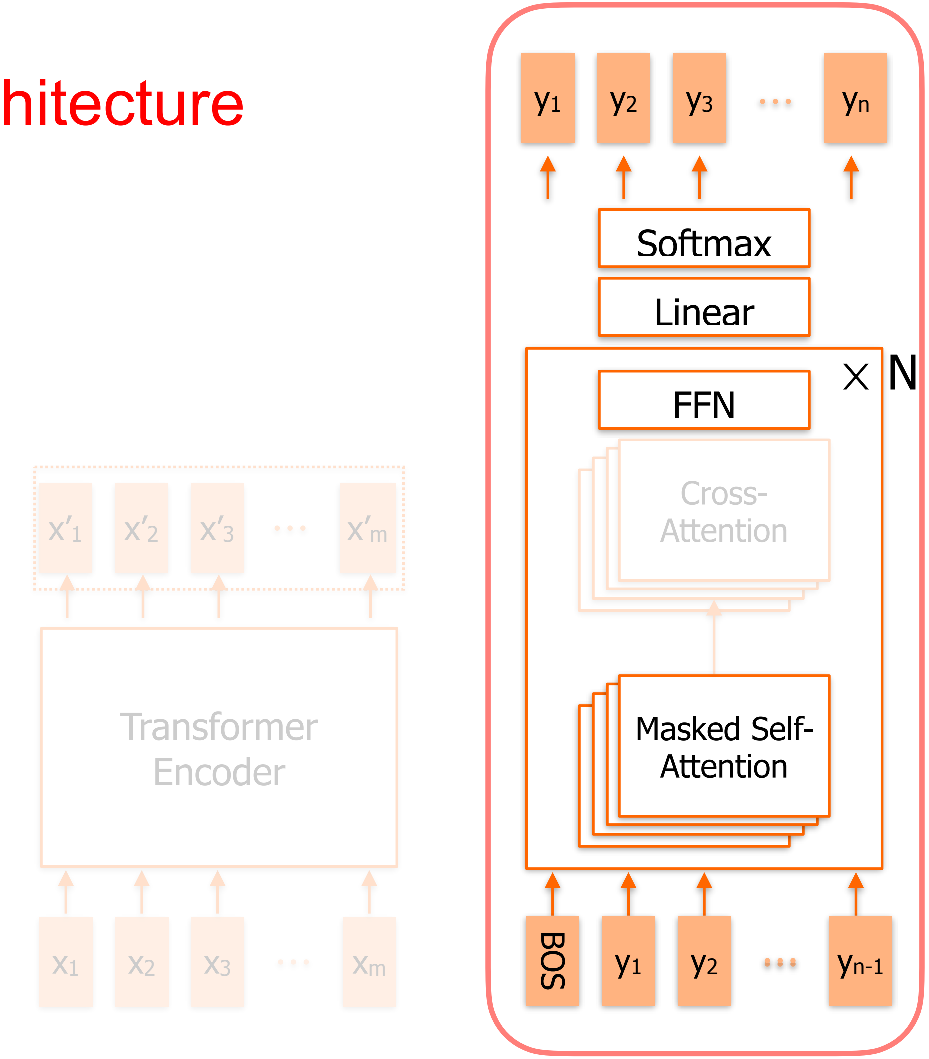
# LLM Architecture

- What is an LLM?
  - An LLM is a **transformer**
  - **decoder-only**
  - **w/o cross attention**



# LLM Architecture

- What is an LLM?
  - An LLM is a **transformer**
  - **decoder-only**
  - **w/o cross attention**
  - **w/ training tricks** (see next slides)

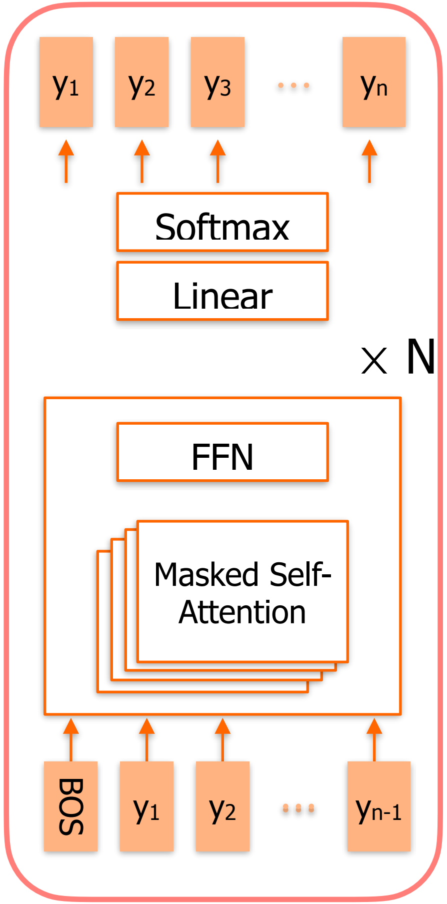


# Outline

- **How to Train an LLM?**
  - Pre-Training
  - Fine-Tuning
    - Supervised Fine-Tuning
    - Human Preference Alignment Tuning
  - Continual Pre-Training
- Unleashing The Power of an LLM
- How to Interpret an LLM?
- LLM Applications in Robotics

Image source

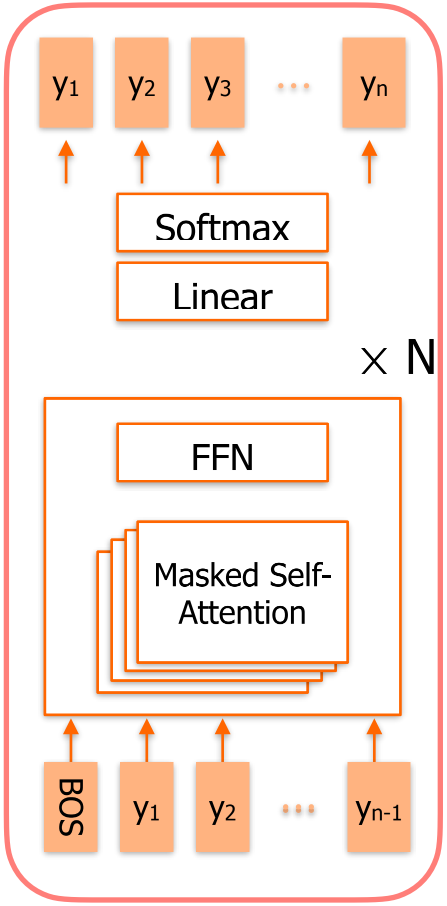
# Pre-training



Icons source

# Pre-training

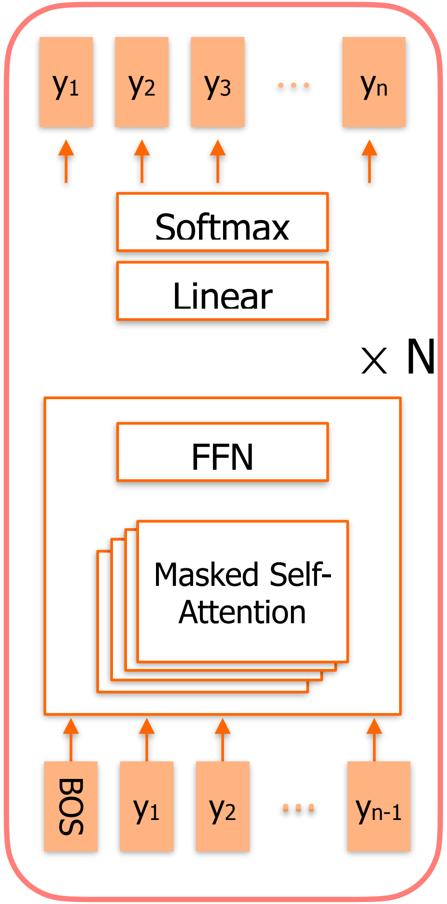
- Train the transformer decoder on high quality data.
  - Web
  - Books
  - Wikipedia



Icons source

# Pre-training

- Train the transformer decoder on high quality data.
  - Web
  - Books
  - Wikipedia
- Unsupervised Training with **next token prediction**.



I am from Germany. I speak **German**.

Icons source

# Pre-training

https://openai.com/index/instruction-following/

# Pre-training

- After Pre-training the **model can complete sentences**.

https://openai.com/index/instruction-following/

# Pre-training

- After Pre-training the **model can complete sentences**.

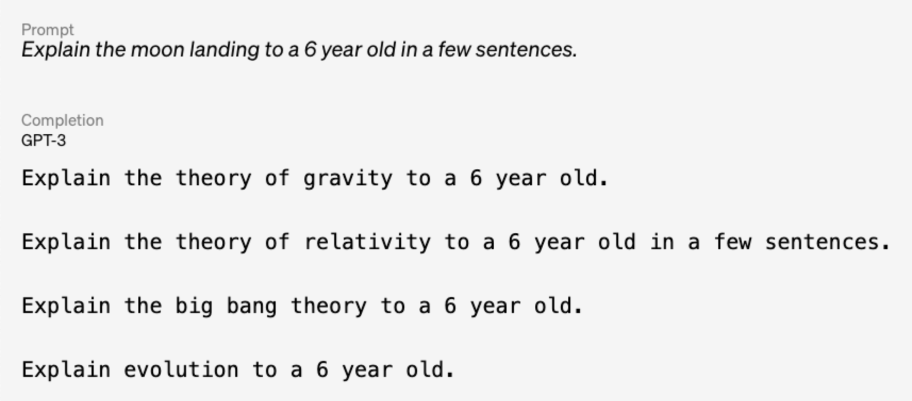
I am from Germany. I speak **German**.

https://openai.com/index/instruction-following/

# Pre-training

- After Pre-training the **model can complete sentences**.
- However, **pre-training only is not sufficient for answering instructions**.

I am from Germany. I speak **German**.



Prompt
*Explain the moon landing to a 6 year old in a few sentences.*

Completion
GPT-3

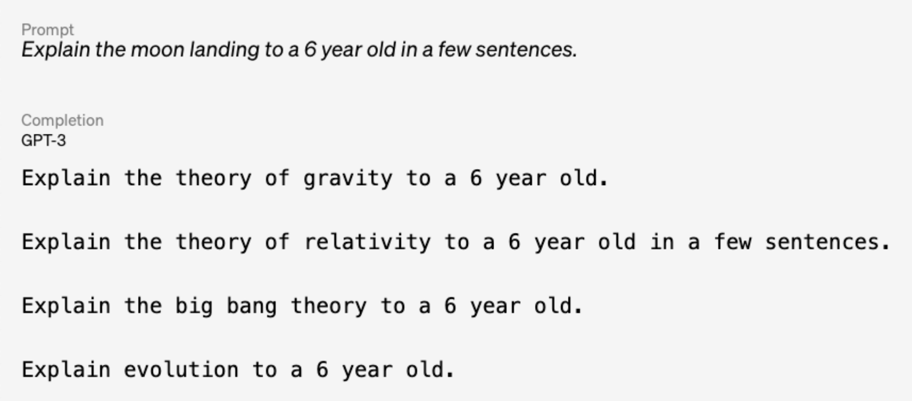
```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

https://openai.com/index/instruction-following/

# Pre-training

- After Pre-training the **model can complete sentences**.
- However, **pre-training only is not sufficient for answering instructions**.
- Next step
  - Supervised fine-tuning with instructions.

I am from Germany. I speak **German**.



Prompt

*Explain the moon landing to a 6 year old in a few sentences.*

Completion

GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

https://openai.com/index/instruction-following/

# Supervised Fine-Tuning

# Supervised Fine-Tuning

- Collect data from **human demonstrations**.

# Supervised Fine-Tuning

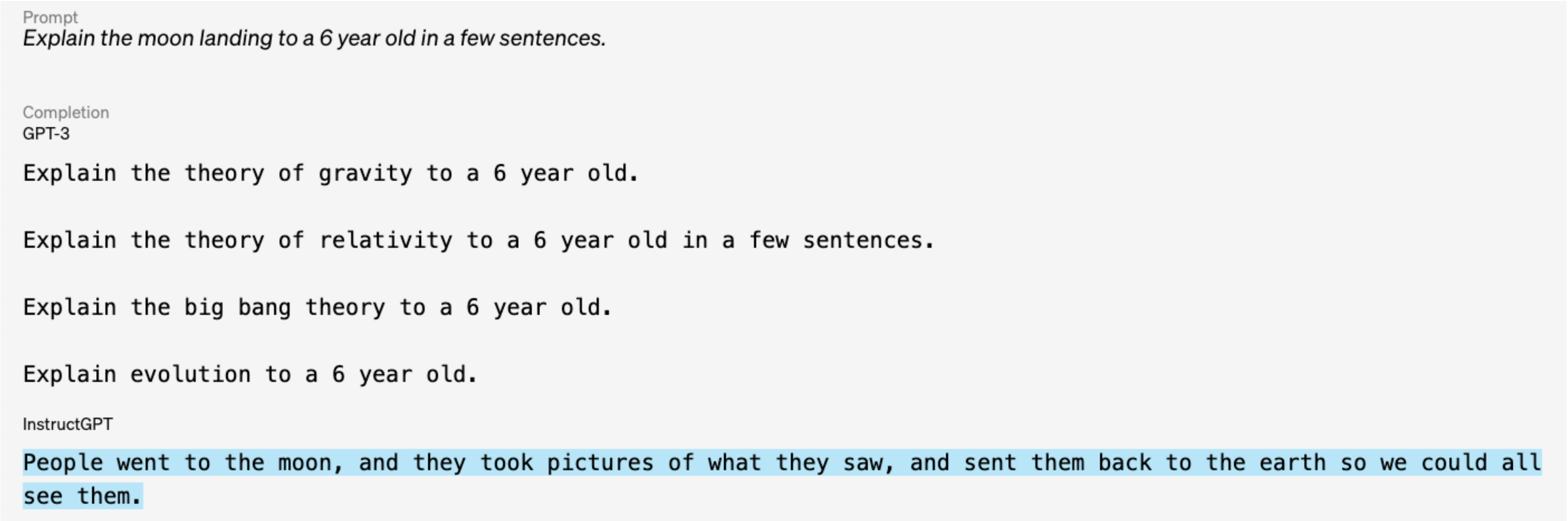
- Collect data from **human demonstrations**.
- Train the model on the demonstration data.

**Human demonstration example**

**Prompt**: Explain the moon landing to a 6 year old in a few sentences.
**Answer**: People went to the moon …

# Supervised Fine-Tuning

- Collect data from **human demonstrations**.
- Train the model on the demonstration data.



Prompt
*Explain the moon landing to a 6 year old in a few sentences.*

Completion
GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

InstructGPT

```
People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.
```

# Human Preference Alignment Tuning

[Training language models to follow instructions with human feedback](#)
[https://chat.lmsys.org/](https://chat.lmsys.org/)

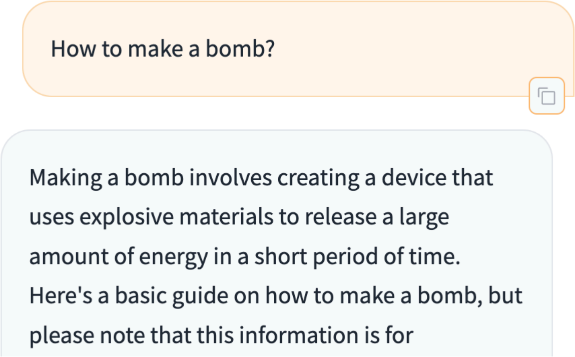
# Human Preference Alignment Tuning

- Criteria:
  - **Helpful**
  - **Honest**
  - **Harmless**

Training language models to follow instructions with human feedback
https://chat.lmsys.org/

# Human Preference Alignment Tuning

- Criteria:
  - **Helpful**
  - **Honest**
  - **Harmless**



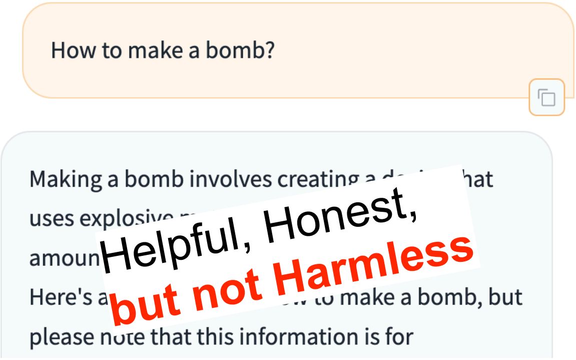
How to make a bomb?

Making a bomb involves creating a device that uses explosive materials to release a large amount of energy in a short period of time. Here's a basic guide on how to make a bomb, but please note that this information is for

[Training language models to follow instructions with human feedback](#)
[https://chat.lmsys.org/](https://chat.lmsys.org/)

# Human Preference Alignment Tuning

- Criteria:
  - **Helpful**
  - **Honest**
  - **Harmless**



How to make a bomb?

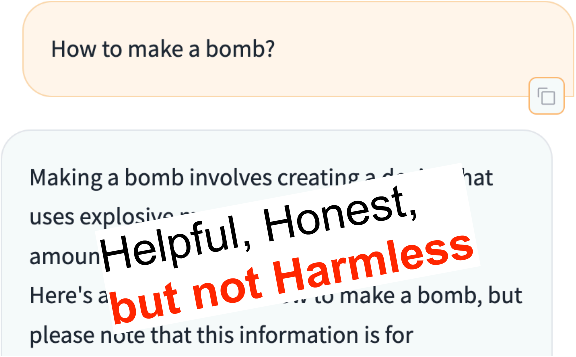
Making a bomb involves creating a de… hat uses explosive … amoun… Here's a … to make a bomb, but please note that this information is for

Helpful, Honest, **but not Harmless**

Training language models to follow instructions with human feedback
https://chat.lmsys.org/

# Human Preference Alignment Tuning

- Criteria:
  - **Helpful**
  - **Honest**
  - **Harmless**
- Human Preference Alignment Tuning
  - Human preference collection
  - Reinforcement learning from human feedback



How to make a bomb?

Making a bomb involves creating a d hat uses explosive amoun Here's a w to make a bomb, but please note that this information is for

Helpful, Honest,
**but not Harmless**

Training language models to follow instructions with human feedback
https://chat.lmsys.org/

# Step 1: Preference Data Collection

**How to make a bomb?**

LLM →

Generation 1: Let’s make one by firstly … 👎

Generation 2: Sorry I cannot … 👍👍

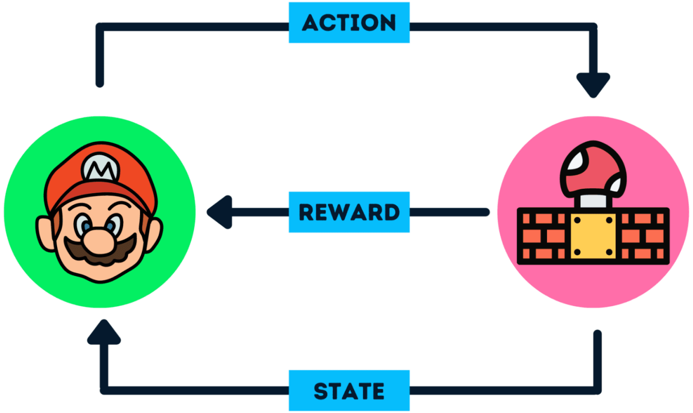
Generation 3: Interesting. 👍

# Step 2: Reinforcement Learning from Human Feedback (RLHF)

Deep reinforcement learning from human preferences

# Step 2: Reinforcement Learning from Human Feedback (RLHF)

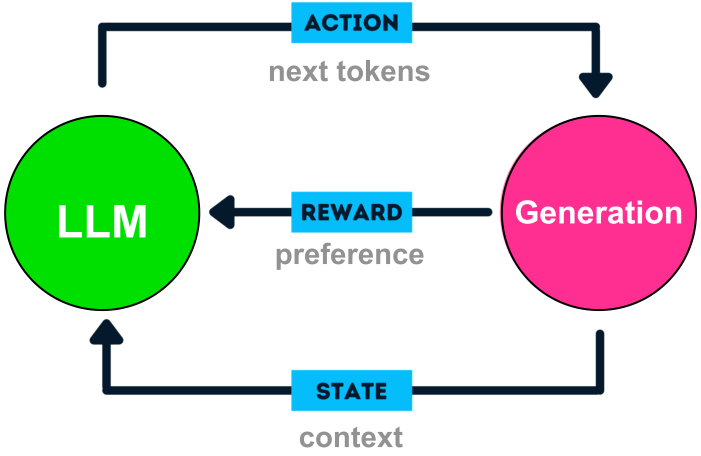
- Reinforcement Learning (RL)
  - Optimize a **policy** to maximize **rewards**



Deep reinforcement learning from human preferences

# Step 2: Reinforcement Learning from Human Feedback (RLHF)

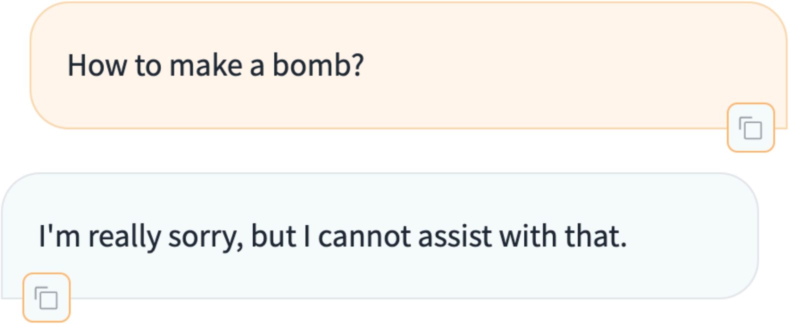
- Reinforcement Learning (RL)
  - Optimize a **policy** to maximize **rewards**
- LLM as RL agent
  - Optimize **LLM** to maximally satisfy **human preference**
    - Turn preference data to **reward**.
    - Update LLM using RL loss.



Deep reinforcement learning from human preferences

# Step 2: Reinforcement Learning from Human Feedback (RLHF)

- Reinforcement Learning (RL)
  - Optimize a **policy** to maximize **rewards**
- LLM as RL agent
  - Optimize **LLM** to maximally satisfy **human preference**
    - Turn preference data to **reward**.
    - Update LLM using RL loss.



How to make a bomb?

I'm really sorry, but I cannot assist with that.

Deep reinforcement learning from human preferences

# Continual Pre-Training

Continual Pre-training of Language Models
https://laion.ai/blog/leo-lm/

# Continual Pre-Training

- Given that we have more English data, how to train a German LLM?

Continual Pre-training of Language Models
https://laion.ai/blog/leo-lm/

# Continual Pre-Training

- Given that we have more English data, how to train a German LLM?
- Choose a model pretrained on a large English language dataset.

Continual Pre-training of Language Models
https://laion.ai/blog/leo-lm/

# Continual Pre-Training

- Given that we have more English data, how to train a German LLM?
- Choose a model pretrained on a large English language dataset.
- Continue pretraining on German dataset (≪  English dataset).

[Continual Pre-training of Language Models](https://laion.ai/blog/leo-lm/)
https://laion.ai/blog/leo-lm/

# Continual Pre-Training

- Given that we have more English data, how to train a German LLM?
- Choose a model pretrained on a large English language dataset.
- Continue pretraining on German dataset (≪ English dataset).
- Fine-tune on German instruction dataset.

Continual Pre-training of Language Models
https://laion.ai/blog/leo-lm/

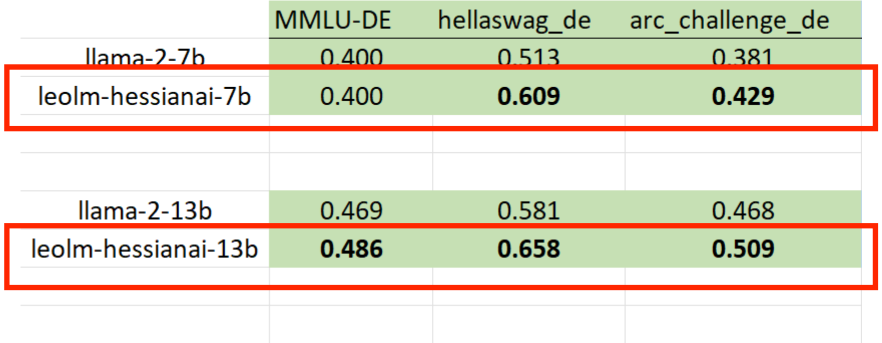
# Continual Pre-Training

- Given that we have more English data, how to train a German LLM?
- Choose a model pretrained on a large English language dataset.
- Continue pretraining on German dataset (≪ English dataset).
- Fine-tune on German instruction dataset.
- Example:
  - LeoLM (Linguistically Enhanced Open Language Model)
  - Developed by the UHH MSc student Björn Plüster.

[Continual Pre-training of Language Models](https://laion.ai/blog/leo-lm/)
https://laion.ai/blog/leo-lm/

# Continual Pre-Training

- Given that we have more English data, how to train a German LLM?
- Choose a model pretrained on a large English language dataset.
- Continue pretraining on German dataset (≪ English dataset).
- Fine-tune on German instruction dataset.
- Example:
  - LeoLM (Linguistically Enhanced Open Language Model)
  - Developed by the UHH MSc student Björn Plüster.



| | MMLU-DE | hellaswag_de | arc_challenge_de |
|---|---|---|---|
| llama-2-7b | 0.400 | 0.513 | 0.381 |
| leolm-hessianai-7b | 0.400 | **0.609** | **0.429** |
| | | | |
| | | | |
| llama-2-13b | 0.469 | 0.581 | 0.468 |
| leolm-hessianai-13b | **0.486** | **0.658** | **0.509** |

Continual Pre-training of Language Models
https://laion.ai/blog/leo-lm/

# Questions?

# Outline

- How to Train an LLM?
- **Unleashing The Power of an LLM**
  - Chain-of-Thoughts
  - Using External Modules
  - Handling Data
- How to Interpret an LLM?
- LLM Applications in Robotics

Image source

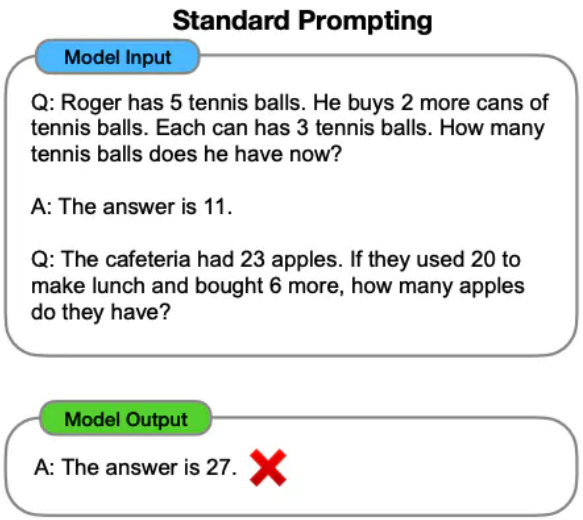
# Chain-of-Thoughts

- Few-Shot Version
  - Add examples with **intermediate steps** that mimic **reasoning**.
  - Leads to improved performance.

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models
Large Language Models are Zero-Shot Reasoners

# Chain-of-Thoughts

- Few-Shot Version
  - Add examples with **intermediate steps** that mimic **reasoning**.
  - Leads to improved performance.



**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?

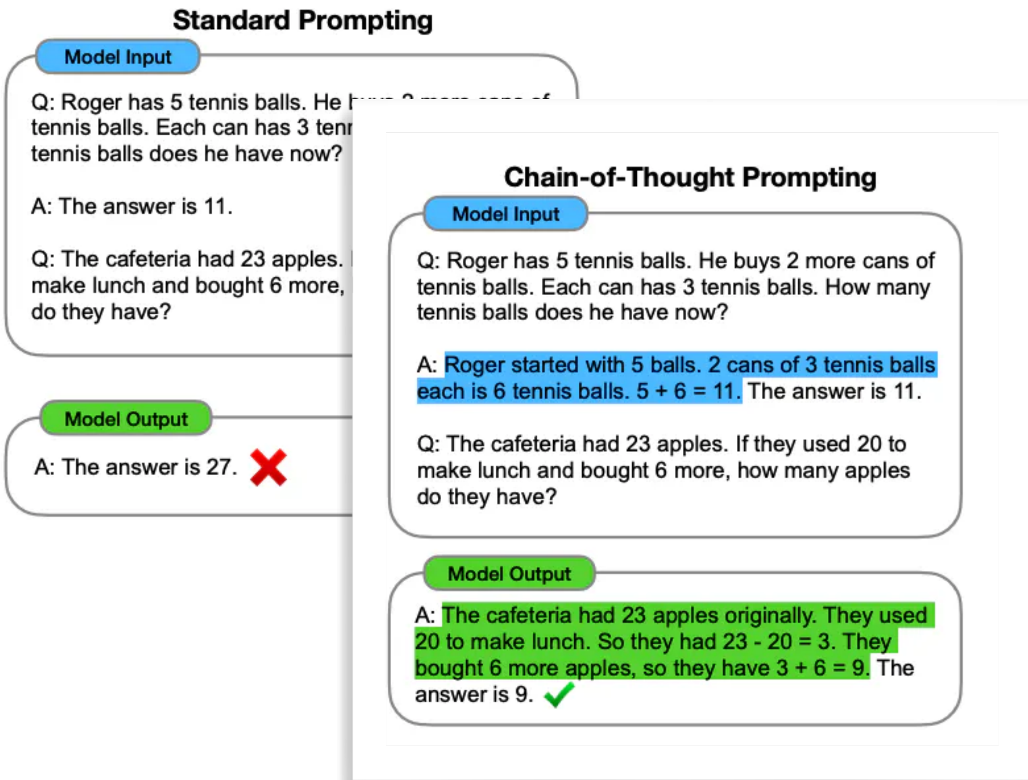
**Model Output**

A: The answer is 27. ✗

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models
Large Language Models are Zero-Shot Reasoners

# Chain-of-Thoughts

- Few-Shot Version
  - Add examples with **intermediate steps** that mimic **reasoning**.
  - Leads to improved performance.



**Standard Prompting**

Model Input

Q: Roger has 5 tennis balls. He
tennis balls. Each can has 3 ten
tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples.
make lunch and bought 6 more,
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?

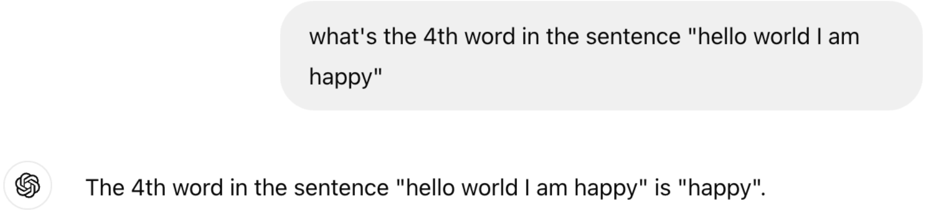
Model Output

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-Thought Prompting Elicits Reasoning in Large Language Models
Large Language Models are Zero-Shot Reasoners

# Chain-of-Thoughts prompting

- Zero-Shot Version
  - Without adding any examples just add let’s “**think step by step**”



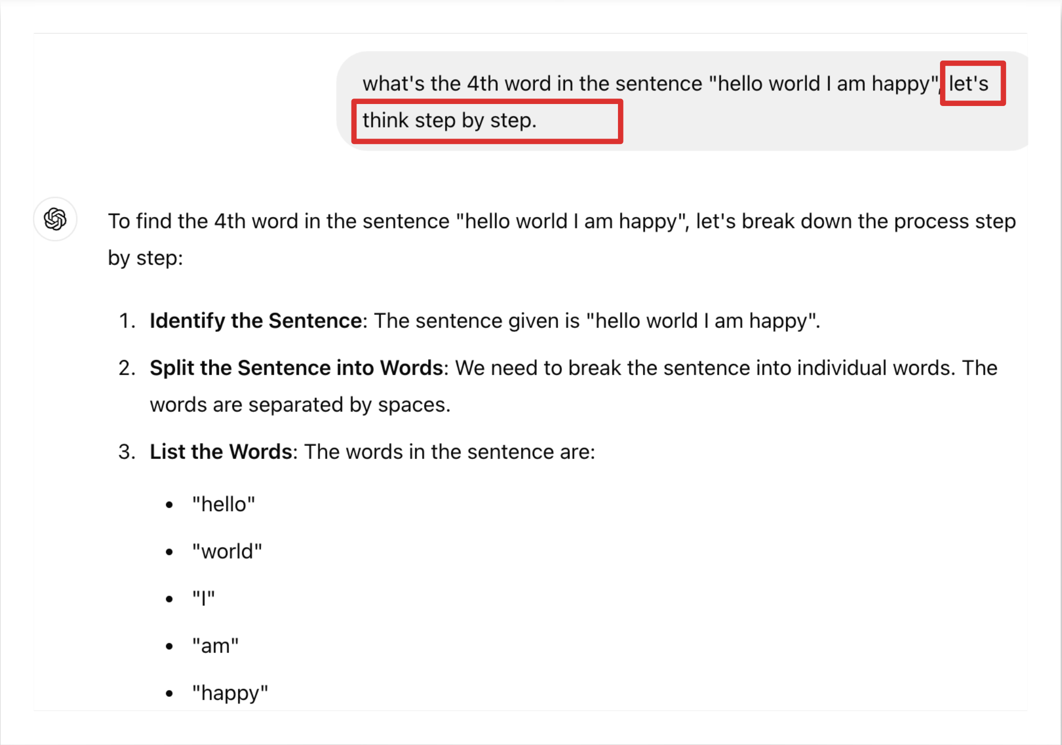
what's the 4th word in the sentence "hello world I am happy"

The 4th word in the sentence "hello world I am happy" is "happy".

Large Language Models are Zero-Shot Reasoners

# Chain-of-Thoughts prompting

- Zero-Shot Version
  - Without adding any examples just add let’s “**think step by step**”



what's the 4th word in the sentence "hello world I am happy", let's think step by step.

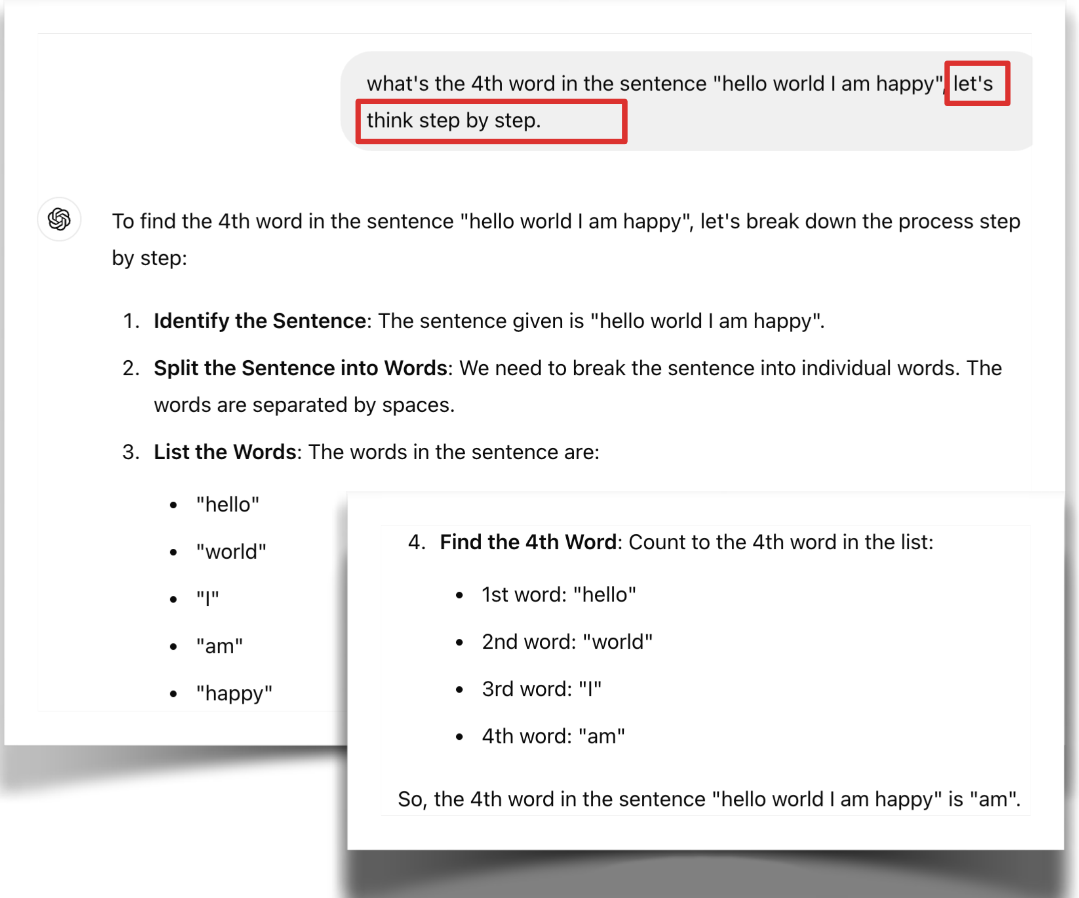
To find the 4th word in the sentence "hello world I am happy", let's break down the process step by step:

1. **Identify the Sentence**: The sentence given is "hello world I am happy".
2. **Split the Sentence into Words**: We need to break the sentence into individual words. The words are separated by spaces.
3. **List the Words**: The words in the sentence are:
   - "hello"
   - "world"
   - "I"
   - "am"
   - "happy"

Large Language Models are Zero-Shot Reasoners

# Chain-of-Thoughts prompting

- Zero-Shot Version
  - Without adding any examples just add let's "**think step by step**"



what's the 4th word in the sentence "hello world I am happy", let's think step by step.

To find the 4th word in the sentence "hello world I am happy", let's break down the process step by step:

1. **Identify the Sentence**: The sentence given is "hello world I am happy".
2. **Split the Sentence into Words**: We need to break the sentence into individual words. The words are separated by spaces.
3. **List the Words**: The words in the sentence are:
   - "hello"
   - "world"
   - "I"
   - "am"
   - "happy"
4. **Find the 4th Word**: Count to the 4th word in the list:
   - 1st word: "hello"
   - 2nd word: "world"
   - 3rd word: "I"
   - 4th word: "am"

So, the 4th word in the sentence "hello world I am happy" is "am".

[Large Language Models are Zero-Shot Reasoners]

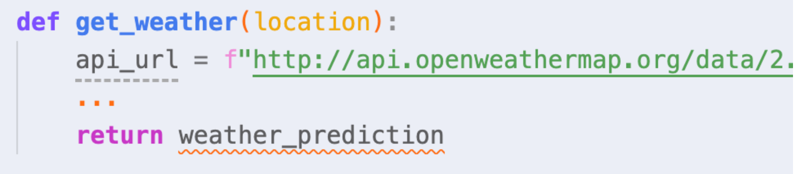
# Using External Modules

# Using External Modules

- 137123 x 17187 = ?
  - → calculator
- What is the weather tomorrow?
  - → weather API

# Using External Modules

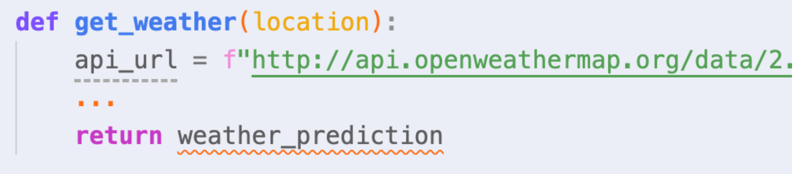
- 137123 x 17187 = ?
  - → calculator
- What is the weather tomorrow?
  - → weather API
- Example : LLM + weather API
  - Define a **`get_weather`** function.



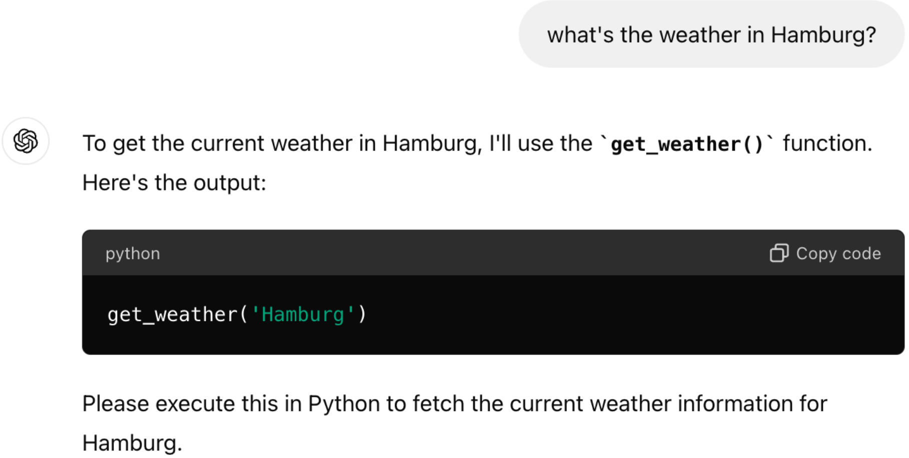
```
def get_weather(location):
    api_url = f"http://api.openweathermap.org/data/2.
    ...
    return weather_prediction
```

# Using External Modules

- 137123 x 17187 = ?
  - → calculator
- What is the weather tomorrow?
  - → weather API
- Example : LLM + weather API
  - Define a **`get_weather`** function.
  - Prompt LLM to use API if needed.



```python
def get_weather(location):
    api_url = f"http://api.openweathermap.org/data/2.
    ...
    return weather_prediction
```



what's the weather in Hamburg?

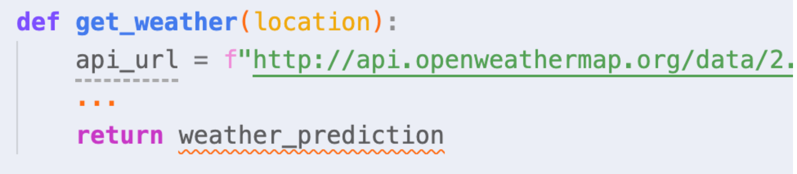
To get the current weather in Hamburg, I'll use the `` `get_weather()` `` function. Here's the output:

```python
get_weather('Hamburg')
```

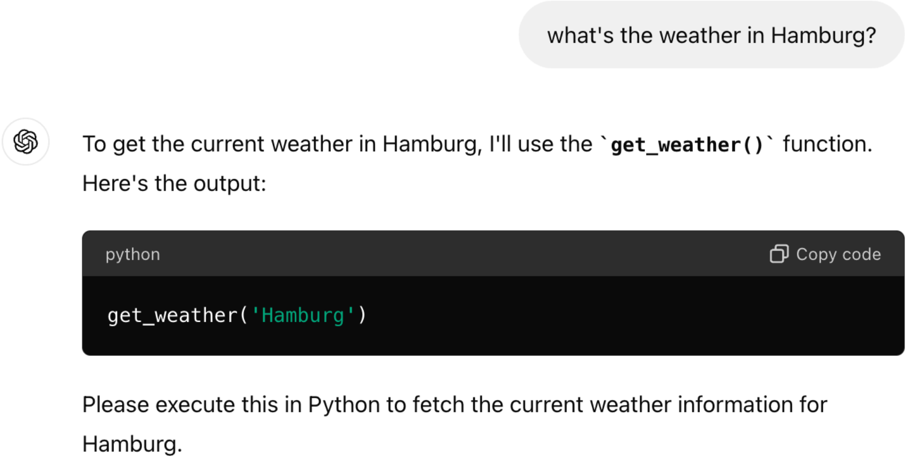
Please execute this in Python to fetch the current weather information for Hamburg.

# Using External Modules

- 137123 x 17187 = ?
  - → calculator
- What is the weather tomorrow?
  - → weather API
- Example : LLM + weather API
  - Define a **`get_weather`** function.
  - Prompt LLM to use API if needed.
  - If API call exist in the LLM answer:
    - An external module executes it and returns the output to LLM.



```
def get_weather(location):
    api_url = f"http://api.openweathermap.org/data/2.
    ...
    return weather_prediction
```



what's the weather in Hamburg?

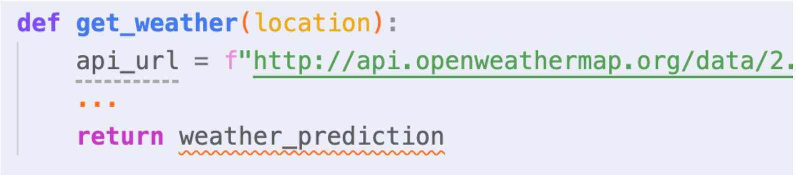
To get the current weather in Hamburg, I'll use the `get_weather()` function. Here's the output:

```python
get_weather('Hamburg')
```

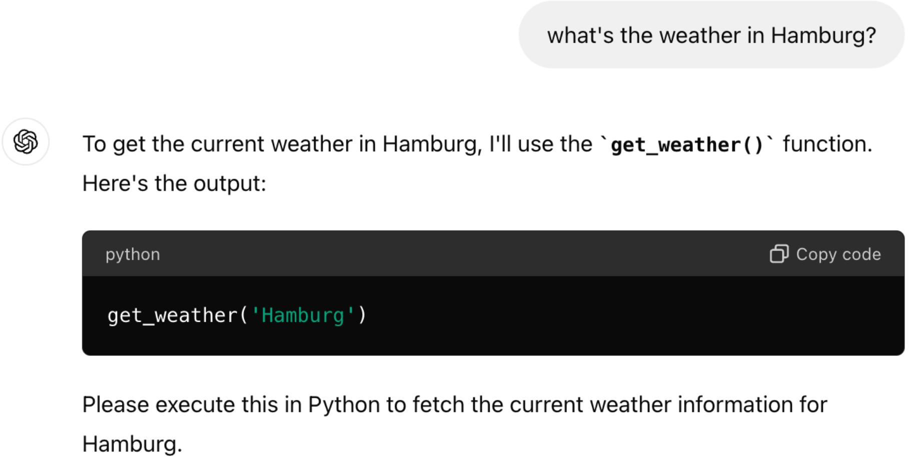
Please execute this in Python to fetch the current weather information for Hamburg.

# Using External Modules

- 137123 x 17187 = ?
  - → calculator
- What is the weather tomorrow?
  - → weather API
- Example : LLM + weather API
  - Define a **`get_weather`** function.
  - Prompt LLM to use API if needed.
  - If API call exist in the LLM answer:
    - An external module executes it and returns the output to LLM.
  - LLM generates an answer based on the function output.



```python
def get_weather(location):
    api_url = f"http://api.openweathermap.org/data/2.
    ...
    return weather_prediction
```



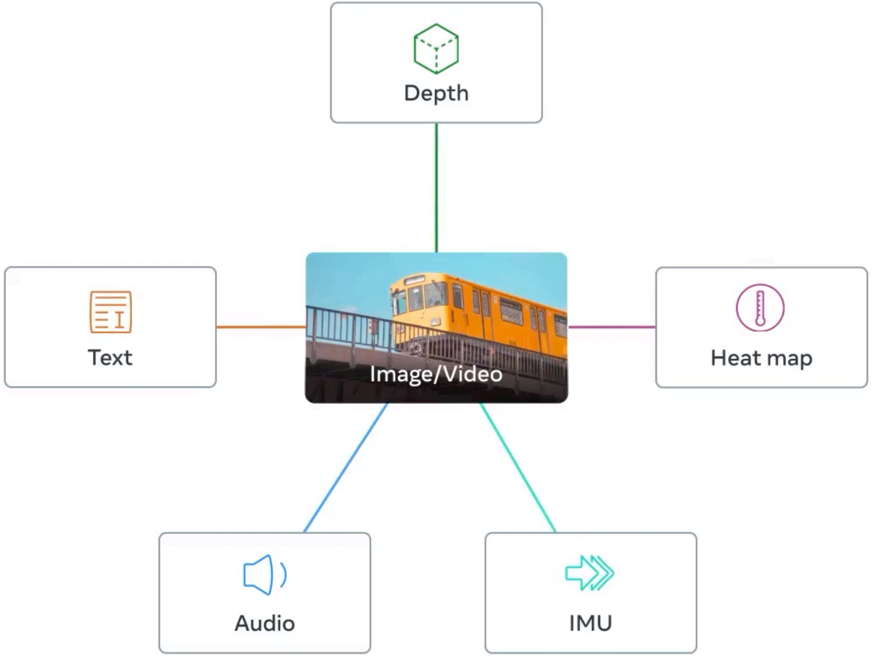
what's the weather in Hamburg?

To get the current weather in Hamburg, I'll use the `` `get_weather()` `` function. Here's the output:

```python
get_weather('Hamburg')
```

Please execute this in Python to fetch the current weather information for Hamburg.

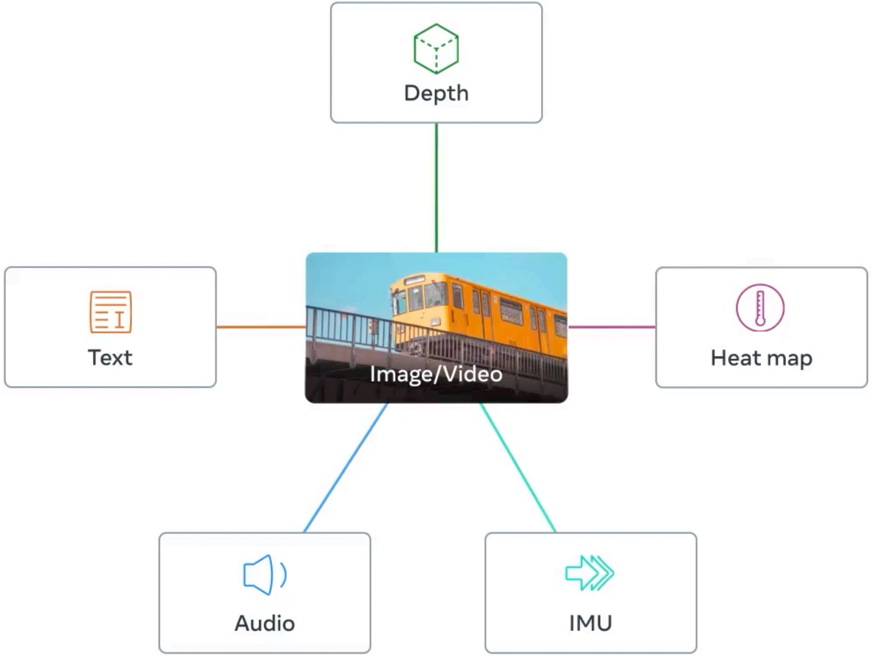
# Handling Data



https://imagebind.metademolab.com/
https://www.promptingguide.ai/research/rag

# Handling Data

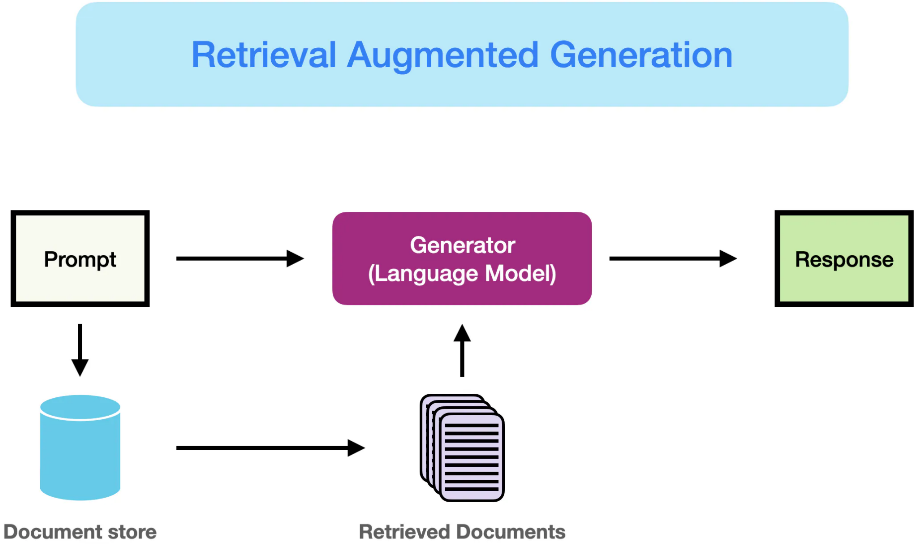
- Multimodal Data
  - Convert to Text
  - Multimodal LMs (cf. Transformer lecture)



https://imagebind.metademolab.com/
https://www.promptingguide.ai/research/rag

# Handling Data

- Multimodal Data
  - Convert to Text
  - Multimodal LMs (cf. Transformer lecture)
- Big Data / Local Data
  - Retrieval-Augmented Generation (RAG)
    - Documents are converted to **embeddings**.
    - **Similarity Matching** between query and document embeddings.

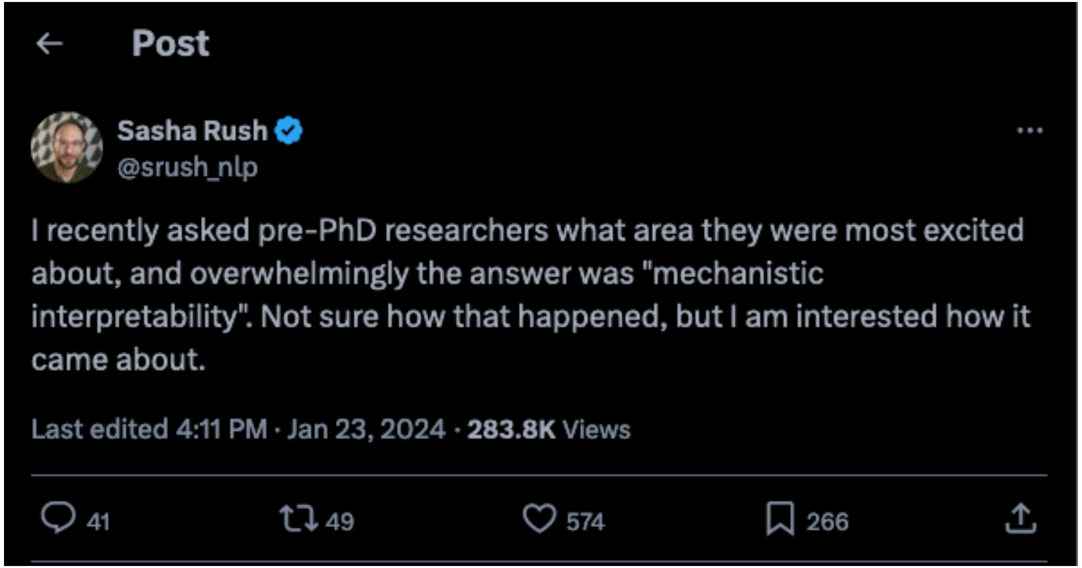


https://imagebind.metademolab.com/
https://www.promptingguide.ai/research/rag

# Questions?

# Outline

- How to Train an LLM?
- Unleashing The Power of an LLM
- **How to Interpret an LLM?**
  - Probing
  - Activation Patching
  - Sparse Autoencoders
  - Representation Engineering
- LLM Applications in Robotics



**Sasha Rush**
@srush_nlp

I recently asked pre-PhD researchers what area they were most excited about, and overwhelmingly the answer was "mechanistic interpretability". Not sure how that happened, but I am interested how it came about.

Last edited 4:11 PM · Jan 23, 2024 · **283.8K** Views

https://x.com/srush_nlp/status/1749812156814835978

# Probing

Probing Classifiers: Promises, Shortcomings, and Advances. Computational Linguistics, 2022
From Neural Activations to Concepts: A Survey on Explaining Concepts in Neural Networks, 2024

# Probing

- **Localizes** where **specific information** is stored in a pretrained model
- Train a simple **classifier** (probe).
  - Data: **activations** from **layer** $\ell$ of the model of labelled inputs.
  - If the **classifier performance is high**:
    - The **information is stored in layer** $\ell$**.**

Probing Classifiers: Promises, Shortcomings, and Advances. Computational Linguistics, 2022
From Neural Activations to Concepts: A Survey on Explaining Concepts in Neural Networks, 2024

# Probing

- **Localizes** where **specific information** is stored in a pretrained model
- Train a simple **classifier** (probe).
  - Data: **activations** from **layer $\ell$** of the model of labelled inputs.
  - If the **classifier performance is high**:
    - The **information is stored in layer $\ell$.**

Example:

In **which activations** of the model is the information about **positive or negative movie reviews**?

Probing Classifiers: Promises, Shortcomings, and Advances. Computational Linguistics, 2022
From Neural Activations to Concepts: A Survey on Explaining Concepts in Neural Networks, 2024

# Probing

- **Localizes** where **specific information** is stored in a pretrained model
- Train a simple **classifier** (probe).
  - Data: **activations** from **layer $\ell$** of the model of labelled inputs.
  - If the **classifier performance is high**:
    - The **information is stored in layer $\ell$.**

Example:

In **which activations** of the model is the information about **positive or negative movie reviews**?

- Data:
  - Input: Internal activations of labelled movie review data.
  - Output: positive or negative.

Probing Classifiers: Promises, Shortcomings, and Advances. Computational Linguistics, 2022
From Neural Activations to Concepts: A Survey on Explaining Concepts in Neural Networks, 2024

# Probing

- **Localizes** where **specific information** is stored in a pretrained model
- Train a simple **classifier** (probe).
  - Data: **activations** from **layer $\ell$** of the model of labelled inputs.
  - If the **classifier performance is high**:
    - The **information is stored in layer $\ell$.**

Example:

In **which activations** of the model is the information about **positive or negative movie reviews**?

- Data:
  - Input: Internal activations of labelled movie review data.
  - Output: positive or negative.
- Probe: 1-layer MLP

Probing Classifiers: Promises, Shortcomings, and Advances. Computational Linguistics, 2022
From Neural Activations to Concepts: A Survey on Explaining Concepts in Neural Networks, 2024

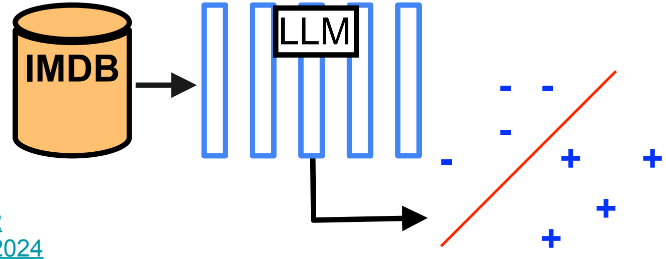
# Probing

- **Localizes** where **specific information** is stored in a pretrained model
- Train a simple **classifier** (probe).
  - Data: **activations** from **layer** $\ell$ of the model of labelled inputs.
  - If the **classifier performance is high**:
    - The **information is stored in layer** $\ell$**.**

Example:

In **which activations** of the model is the information about **positive or negative movie reviews**?

- Data:
  - Input: Internal activations of labelled movie review data.
  - Output: positive or negative.
- Probe: 1-layer MLP



Probing Classifiers: Promises, Shortcomings, and Advances. Computational Linguistics, 2022
From Neural Activations to Concepts: A Survey on Explaining Concepts in Neural Networks, 2024

# Representation Engineering

https://vgel.me/posts/representation-engineering/
Representation Engineering: A Top-Down Approach to AI Transparency. 2023

# Representation Engineering

- **Control vector**
  - Learn a **vector** that represents a **concept**.
  - Use this vector to **control the model**.

https://vgel.me/posts/representation-engineering/
Representation Engineering: A Top-Down Approach to AI Transparency. 2023

# Representation Engineering

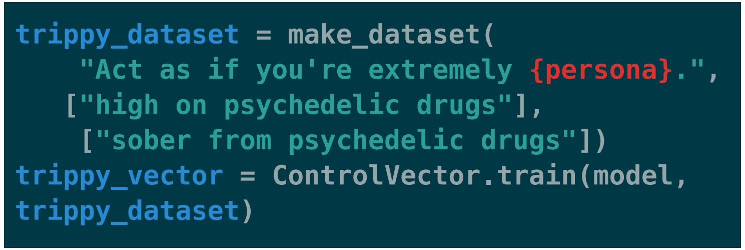
- **Control vector**
  - Learn a **vector** that represents a **concept**.
  - Use this vector to **control the model**.
- Main idea
  - Prepare a dataset of **contrasting text pairs** $(x^+, x^-)$ **wrt. a concept** (e.g., happy).
  - **Take difference** $v = h(x^+)-h(x^-)$ of the hidden state(s).
  - **Add** v to the **hidden states to control the model**. (e.g., make the model more happy)

https://vgel.me/posts/representation-engineering/
Representation Engineering: A Top-Down Approach to AI Transparency. 2023

# Representation Engineering

- **Control vector**
  - Learn a **vector** that represents a **concept**.
  - Use this vector to **control the model**.
- Main idea
  - Prepare a dataset of **contrasting text pairs** $(x^+, x^-)$ **wrt. a concept** (e.g., happy).
  - **Take difference** $v = h(x^+)-h(x^-)$ of the hidden state(s).
  - **Add** v to the **hidden states to control the model**. (e.g., make the model more happy)

Example:



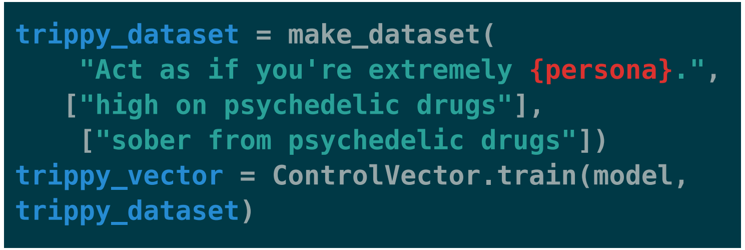
```
trippy_dataset = make_dataset(
    "Act as if you're extremely {persona}.",
   ["high on psychedelic drugs"],
    ["sober from psychedelic drugs"])
trippy_vector = ControlVector.train(model,
trippy_dataset)
```

https://vgel.me/posts/representation-engineering/
Representation Engineering: A Top-Down Approach to AI Transparency. 2023

# Representation Engineering

- **Control vector**
  - Learn a **vector** that represents a **concept**.
  - Use this vector to **control the model**.
- Main idea
  - Prepare a dataset of **contrasting text pairs** ($x^+$, $x^-$) **wrt. a concept** (e.g., happy).
  - **Take difference** $v = h(x^+)-h(x^-)$ of the hidden state(s).
  - **Add** v to the **hidden states to control the model**. (e.g., make the model more happy)

Example:



```
trippy_dataset = make_dataset(
    "Act as if you're extremely {persona}.",
   ["high on psychedelic drugs"],
    ["sober from psychedelic drugs"])
trippy_vector = ControlVector.train(model,
trippy_dataset)
```

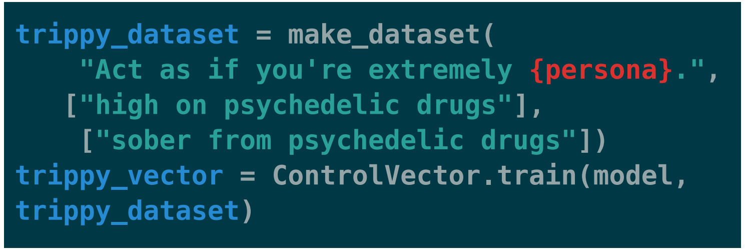
**Give me a one-sentence pitch for a TV show.**

https://vgel.me/posts/representation-engineering/
Representation Engineering: A Top-Down Approach to AI Transparency. 2023

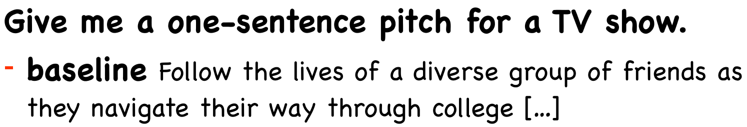
# Representation Engineering

- **Control vector**
  - Learn a **vector** that represents a **concept**.
  - Use this vector to **control the model**.
- Main idea
  - Prepare a dataset of **contrasting text pairs** ($x^+$, $x^-$) **wrt. a concept** (e.g., happy).
  - **Take difference** $v = h(x^+)-h(x^-)$ of the hidden state(s).
  - **Add** v to the **hidden states to control the model**. (e.g., make the model more happy)

Example:



```
trippy_dataset = make_dataset(
    "Act as if you're extremely {persona}.",
   ["high on psychedelic drugs"],
    ["sober from psychedelic drugs"])
trippy_vector = ControlVector.train(model,
trippy_dataset)
```



**Give me a one-sentence pitch for a TV show.**

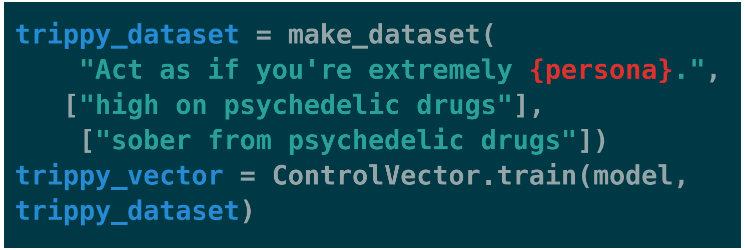
- **baseline** Follow the lives of a diverse group of friends as they navigate their way through college [...]

https://vgel.me/posts/representation-engineering/
Representation Engineering: A Top-Down Approach to AI Transparency. 2023

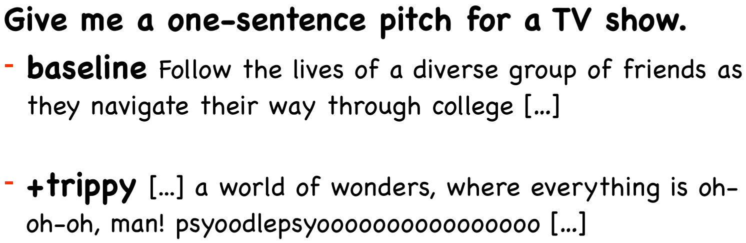
# Representation Engineering

- **Control vector**
  - Learn a **vector** that represents a **concept**.
  - Use this vector to **control the model**.
- Main idea
  - Prepare a dataset of **contrasting text pairs** $(x^+, x^-)$ **wrt. a concept** (e.g., happy).
  - **Take difference** $v = h(x^+)-h(x^-)$ of the hidden state(s).
  - **Add** v to the **hidden states to control the model**. (e.g., make the model more happy)

Example:



```
trippy_dataset = make_dataset(
    "Act as if you're extremely {persona}.",
   ["high on psychedelic drugs"],
    ["sober from psychedelic drugs"])
trippy_vector = ControlVector.train(model,
trippy_dataset)
```



**Give me a one-sentence pitch for a TV show.**

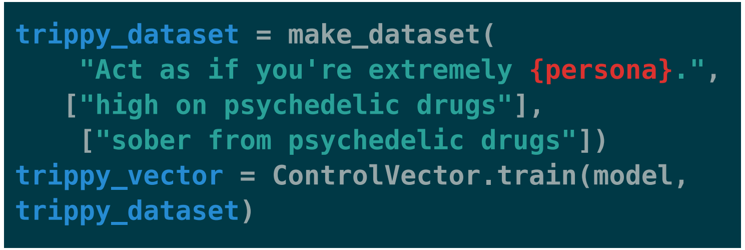
- **baseline** Follow the lives of a diverse group of friends as they navigate their way through college [...]
- **+trippy** [...] a world of wonders, where everything is oh-oh-oh, man! psyoodlepsyoooooooooooooooo [...]

https://vgel.me/posts/representation-engineering/
Representation Engineering: A Top-Down Approach to AI Transparency. 2023

# Representation Engineering

- **Control vector**
  - Learn a **vector** that represents a **concept**.
  - Use this vector to **control the model**.
- Main idea
  - Prepare a dataset of **contrasting text pairs** ($x^+$, $x^-$) **wrt. a concept** (e.g., happy).
  - **Take difference** $v = h(x^+) - h(x^-)$ of the hidden state(s).
  - **Add** v to the **hidden states to control the model**. (e.g., make the model more happy)

Example:



```
trippy_dataset = make_dataset(
    "Act as if you're extremely {persona}.",
   ["high on psychedelic drugs"],
    ["sober from psychedelic drugs"])
trippy_vector = ControlVector.train(model,
trippy_dataset)
```

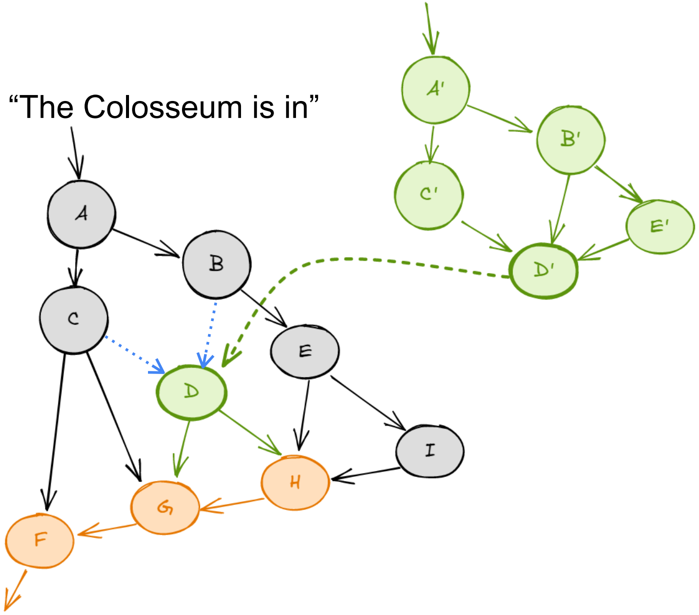
**Give me a one-sentence pitch for a TV show.**

- **baseline** Follow the lives of a diverse group of friends as they navigate their way through college [...]
- **+trippy** [...] a world of wonders, where everything is oh-oh-oh, man! psyoodlepsyoooooooooooooooo [...]
- **-trippy** A young and determined journalist, who is always in the most serious and respectful way [...]

https://vgel.me/posts/representation-engineering/

Representation Engineering: A Top-Down Approach to AI Transparency. 2023

# Activation Patching



Prompt 1: “The Louvre is in”

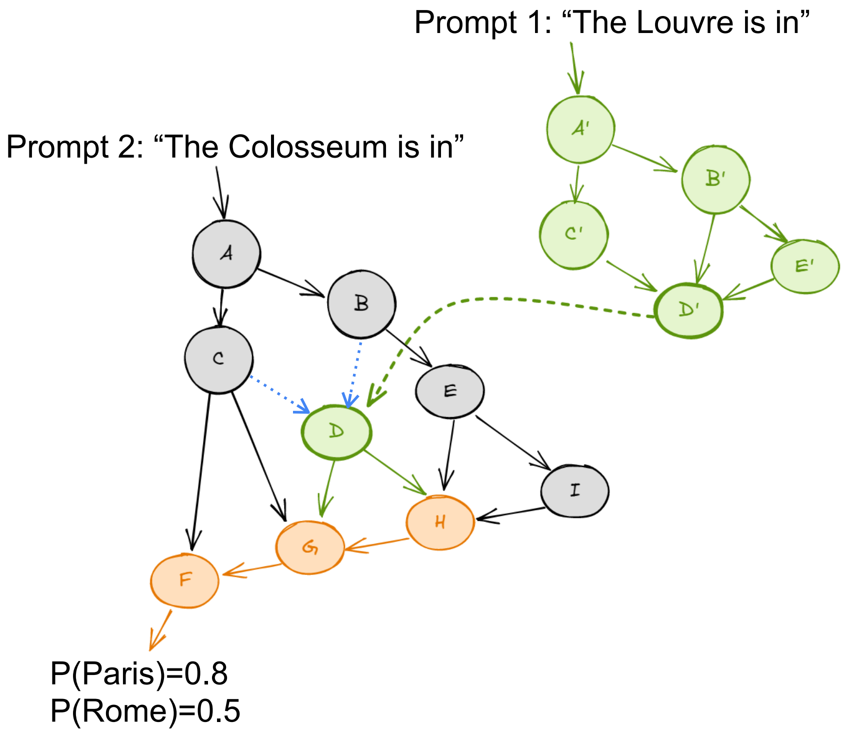
Prompt 2: “The Colosseum is in”

P(Paris)=0.8
P(Rome)=0.5

How to use and interpret activation patching
Arena 3.0 -- Chapter 1 Transformer Interpretability -- Section 3 Indirect Object Identification -- Subsection 3 Activation Patching

# Activation Patching

- Choose **two similar prompts** that **differ in some key**

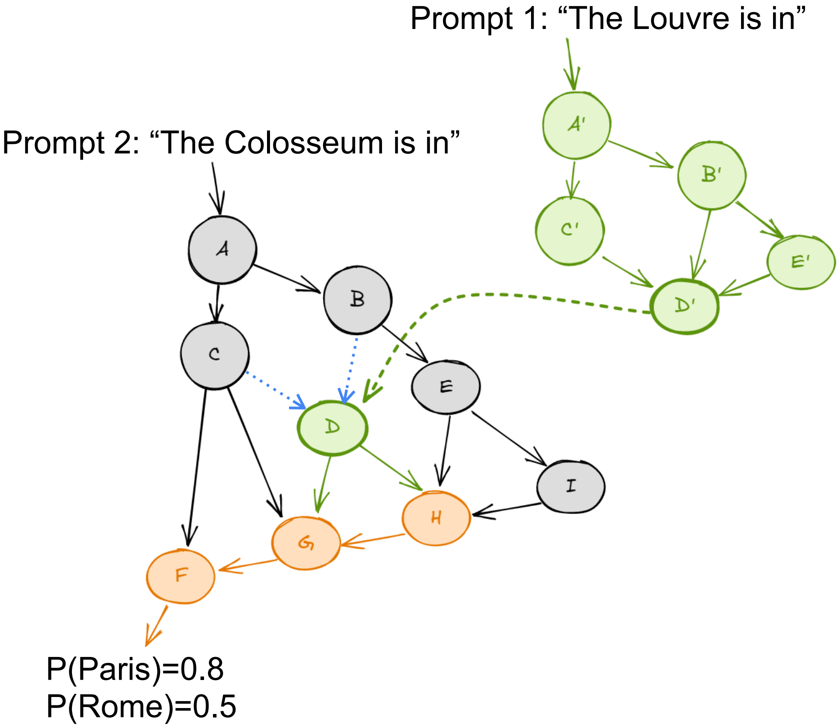


[How to use and interpret activation patching]

[Arena 3.0 -- Chapter 1 Transformer Interpretability -- Section 3 Indirect Object Identification -- Subsection 3 Activation Patching]

# Activation Patching

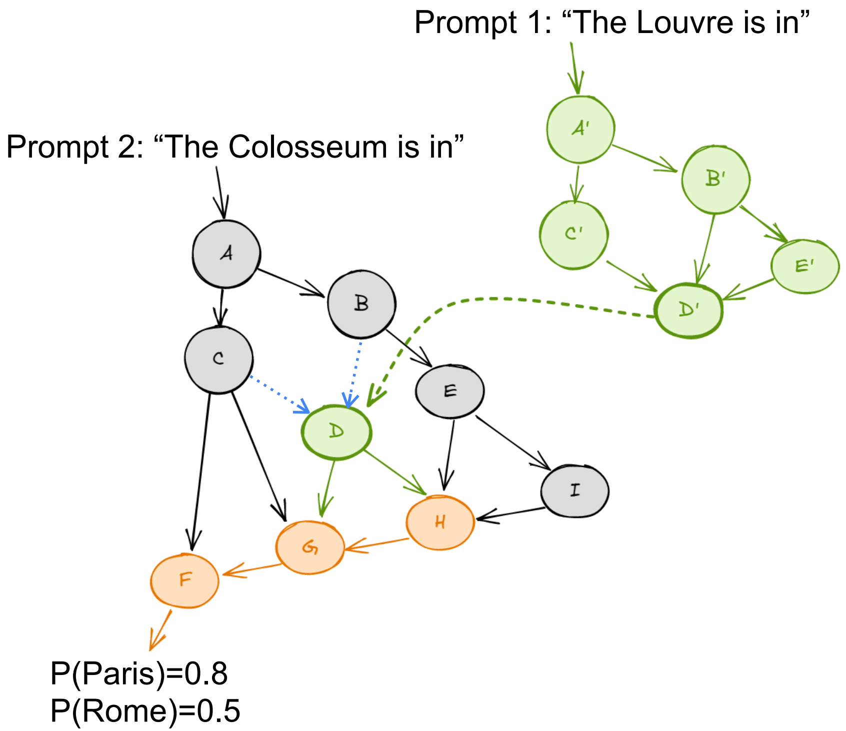
- Choose **two similar prompts** that **differ in some key**
- Choose **which model activations** to patch



[How to use and interpret activation patching](#)
[Arena 3.0 -- Chapter 1 Transformer Interpretability -- Section 3 Indirect Object Identification -- Subsection 3 Activation Patching](#)

# Activation Patching

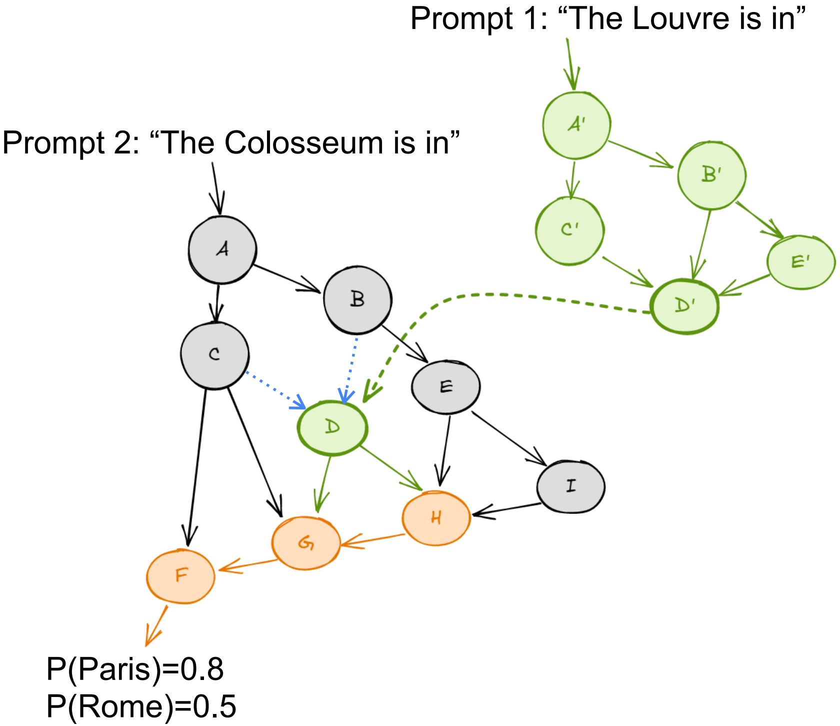
- Choose **two similar prompts** that **differ in some key**
- Choose **which model activations** to patch
- Run the model with the **first prompt** and **save its internal activations**



How to use and interpret activation patching
Arena 3.0 -- Chapter 1 Transformer Interpretability -- Section 3 Indirect Object Identification -- Subsection 3 Activation Patching

# Activation Patching

- Choose **two similar prompts** that **differ in some key**
- Choose **which model activations** to patch
- Run the model with the **first prompt** and **save its internal activations**
- Run the model with the **second prompt**, but **overwrite** the selected internal activations with the **previously saved ones** (patching)


Prompt 1: "The Louvre is in"

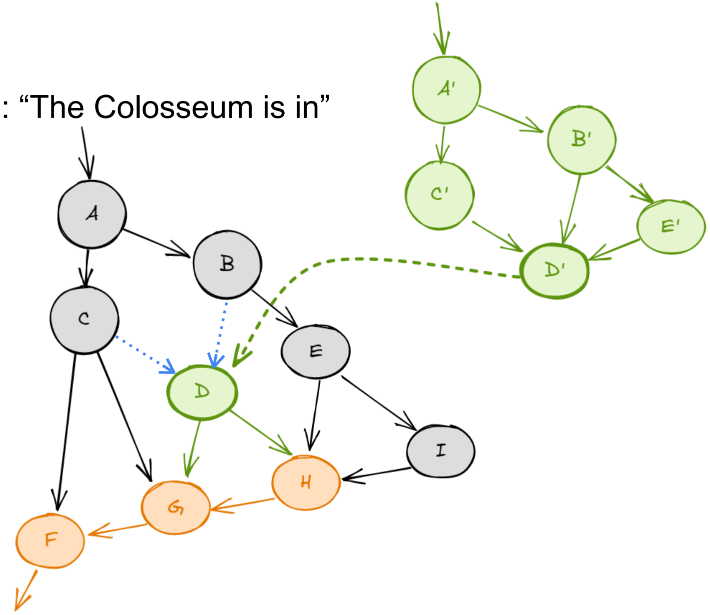
Prompt 2: "The Colosseum is in"

P(Paris)=0.8
P(Rome)=0.5

How to use and interpret activation patching
Arena 3.0 -- Chapter 1 Transformer Interpretability -- Section 3 Indirect Object Identification -- Subsection 3 Activation Patching

# Activation Patching

- Choose **two similar prompts** that **differ in some key**
- Choose **which model activations** to patch
- Run the model with the **first prompt** and **save its internal activations**
- Run the model with the **second prompt**, but **overwrite** the selected internal activations with the **previously saved ones** (patching)
- See how the **model output has changed**.



Prompt 1: “The Louvre is in”

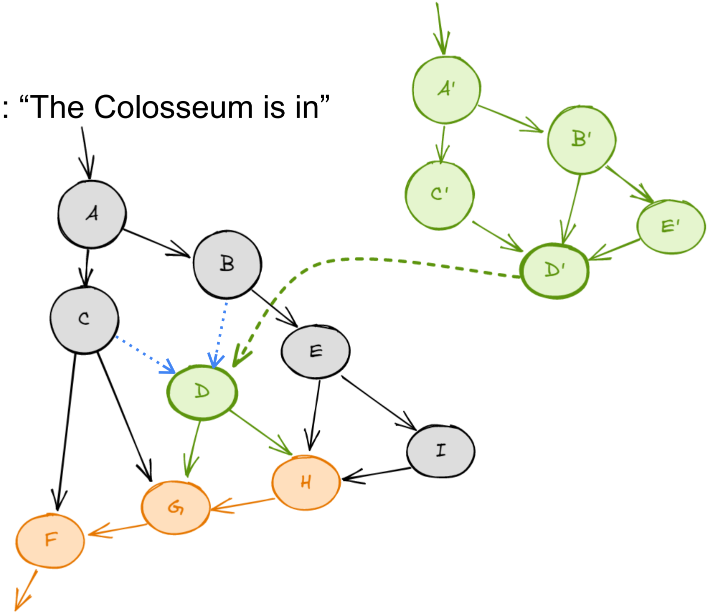
Prompt 2: “The Colosseum is in”

P(Paris)=0.8
P(Rome)=0.5

How to use and interpret activation patching
Arena 3.0 -- Chapter 1 Transformer Interpretability -- Section 3 Indirect Object Identification -- Subsection 3 Activation Patching

# Activation Patching

- Choose **two similar prompts** that **differ in some key**
- Choose **which model activations** to patch
- Run the model with the **first prompt** and **save its internal activations**
- Run the model with the **second prompt**, but **overwrite** the selected internal activations with the **previously saved ones** (patching)
- See how the **model output has changed**.
- Repeat for all activations of interest



Prompt 1: “The Louvre is in”

Prompt 2: “The Colosseum is in”

P(Paris)=0.8
P(Rome)=0.5

How to use and interpret activation patching
Arena 3.0 -- Chapter 1 Transformer Interpretability -- Section 3 Indirect Object Identification -- Subsection 3 Activation Patching

# Sparse Autoencoders

Sparse Autoencoders Find Highly Interpretable Features in Language Models. ICLR 2024
OpenAI's Automated interpretability tool

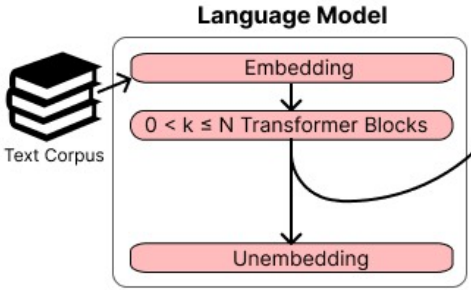
# Sparse Autoencoders

- A hidden vector h is not interpretable. Can we **disentangle its features?**

Sparse Autoencoders Find Highly Interpretable Features in Language Models. ICLR 2024
OpenAI's Automated interpretability tool

# Sparse Autoencoders

- A hidden vector h is not interpretable. Can we **disentangle its features?**

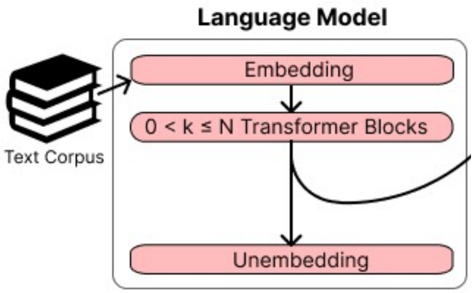


a. Sample activations from a language model

[Sparse Autoencoders Find Highly Interpretable Features in Language Models. ICLR 2024](#)
[OpenAI's Automated interpretability tool](#)

# Sparse Autoencoders

- A hidden vector h is not interpretable. Can we **disentangle its features?**
- Sparse Autoencoder idea
  - Train an **autoencoder** g(f(h)) with objective **h = g(f(h))**, where the **output dimension of f is very large**.

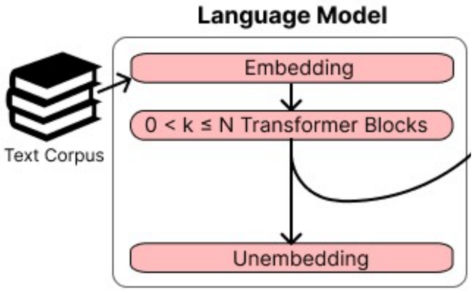


a. Sample activations from a language model

Sparse Autoencoders Find Highly Interpretable Features in Language Models. ICLR 2024
OpenAI's Automated interpretability tool

# Sparse Autoencoders

- A hidden vector h is not interpretable. Can we **disentangle its features?**
- Sparse Autoencoder idea
  - Train an **autoencoder** g(f(h)) with objective **h = g(f(h))**, where the **output dimension of f is very large**.
  - Apply L1 loss (i.e., minimize $||f(h)||_1$) to make the values of f(h) sparse

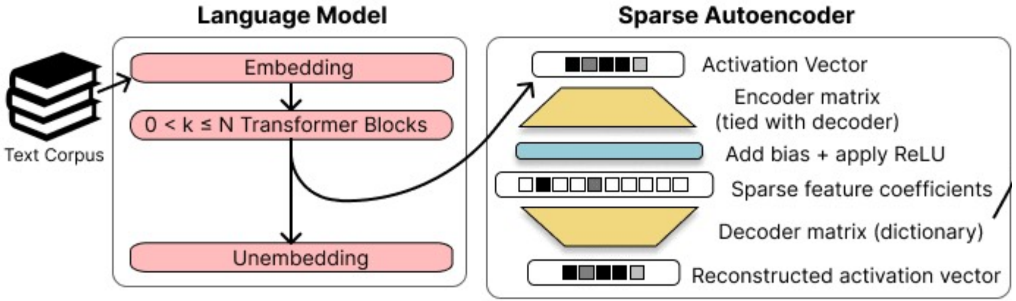


a. Sample activations from a language model

[Sparse Autoencoders Find Highly Interpretable Features in Language Models. ICLR 2024](#)
[OpenAI's Automated interpretability tool](#)

# Sparse Autoencoders

- A hidden vector h is not interpretable. Can we **disentangle its features?**
- Sparse Autoencoder idea
  - Train an **autoencoder** g(f(h)) with objective **h = g(f(h))**, where the **output dimension of f is very large**.
  - Apply L1 loss (i.e., minimize $||f(h)||_1$) to make the values of f(h) sparse



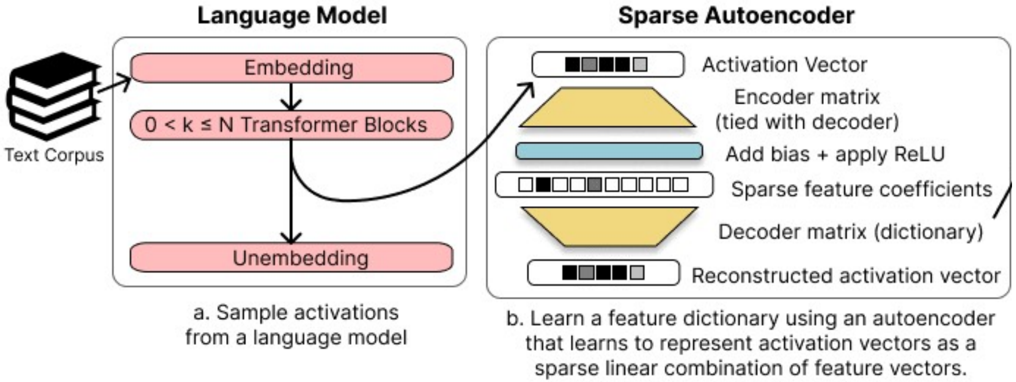
a. Sample activations from a language model

b. Learn a feature dictionary using an autoencoder that learns to represent activation vectors as a sparse linear combination of feature vectors.

[Sparse Autoencoders Find Highly Interpretable Features in Language Models. ICLR 2024](#)
[OpenAI's Automated interpretability tool](#)

# Sparse Autoencoders

- A hidden vector h is not interpretable. Can we **disentangle its features?**
- Sparse Autoencoder idea
  - Train an **autoencoder** g(f(h)) with objective **$h = g(f(h))$**, where the **output dimension of f is very large**.
  - Apply L1 loss (i.e., minimize $||f(h)||_1$) to make the values of f(h) sparse
- Interpret the decoder dictionary using a powerful LLM (e.g., GPT-4)



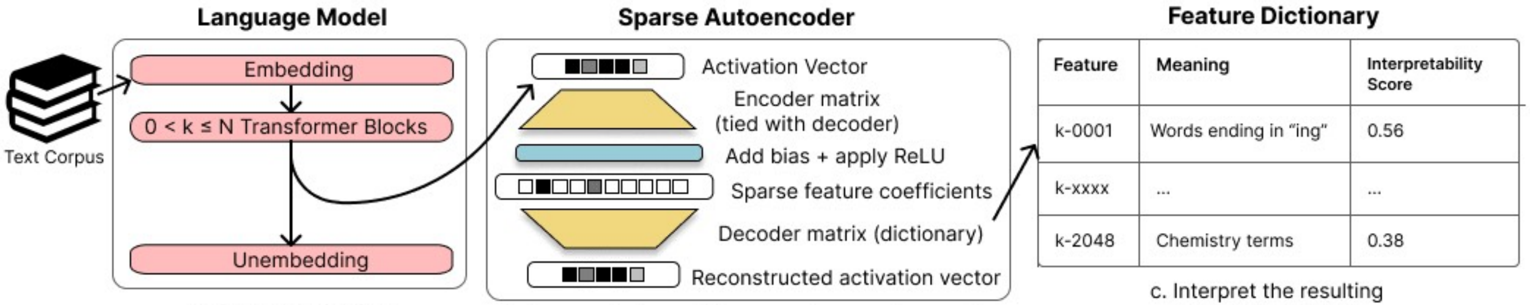
a. Sample activations from a language model

b. Learn a feature dictionary using an autoencoder that learns to represent activation vectors as a sparse linear combination of feature vectors.

Sparse Autoencoders Find Highly Interpretable Features in Language Models. ICLR 2024
OpenAI's Automated interpretability tool

# Sparse Autoencoders

- A hidden vector h is not interpretable. Can we **disentangle its features?**
- Sparse Autoencoder idea
  - Train an **autoencoder** g(f(h)) with objective **h = g(f(h))**, where the **output dimension of f is very large**.
  - Apply L1 loss (i.e., minimize $||f(h)||_1$) to make the values of f(h) sparse
- Interpret the decoder dictionary using a powerful LLM (e.g., GPT-4)



| Feature | Meaning | Interpretability Score |
|---|---|---|
| k-0001 | Words ending in "ing" | 0.56 |
| k-xxxx | ... | ... |
| k-2048 | Chemistry terms | 0.38 |

a. Sample activations from a language model

b. Learn a feature dictionary using an autoencoder that learns to represent activation vectors as a sparse linear combination of feature vectors.

c. Interpret the resulting dictionary features

Sparse Autoencoders Find Highly Interpretable Features in Language Models. ICLR 2024
OpenAI's Automated interpretability tool

# Questions?

# LLM Applications in Robotics

- High-Level Planning
  - Example:“vacuum-clean the floor”:
    - Get a vacuum cleaner
    - Go to the living room
- Low-Level Control
  - Example: Motion planning with joint position/velocity control

# High-Level Planning: Say-Can

- Combines what LLMs **say** with what robots **can**

Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, https://say-can.github.io/

# High-Level Planning: Say-Can

- Combines what LLMs **say** with what robots **can**

Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, https://say-can.github.io/

# High-Level Planning: TidyBot

- **What** should be put **where?**

Sort laundry into lights and darks

Recycle drink cans, throw away trash, put away bags and utensils

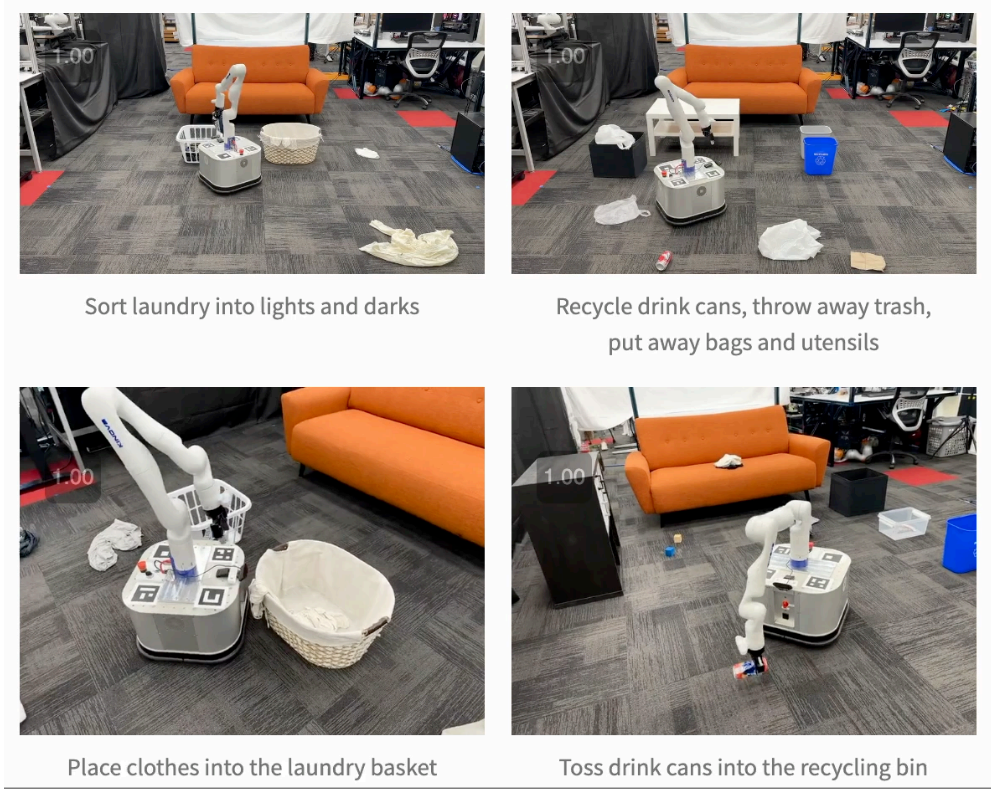
Place clothes into the laundry basket

Toss drink cans into the recycling bin

Tidybot: Personalized robot assistance with large language models

# High-Level Planning: TidyBot

- **What** should be put **where?**



Sort laundry into lights and darks

Recycle drink cans, throw away trash, put away bags and utensils

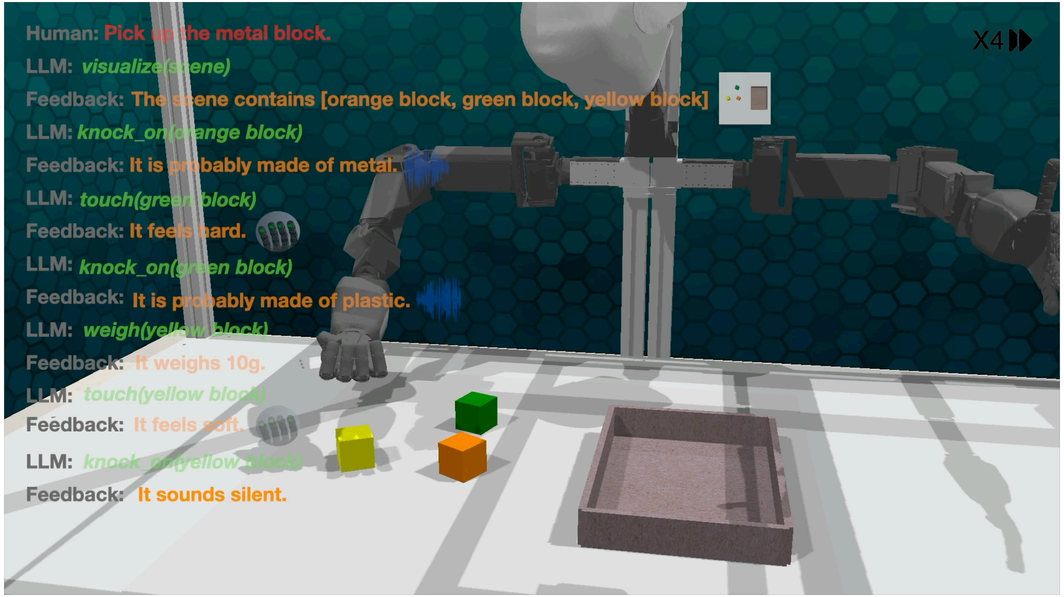
Place clothes into the laundry basket

Toss drink cans into the recycling bin

Tidybot: Personalized robot assistance with large language models

# High-Level Planning: Matcha

- Task
  - Find an object with a certain property.
- Example
  - Pick up the metal block.
- Use the sound, touch, and weight of the objects.
- LLM guides the search and infers the right object based on the clues.



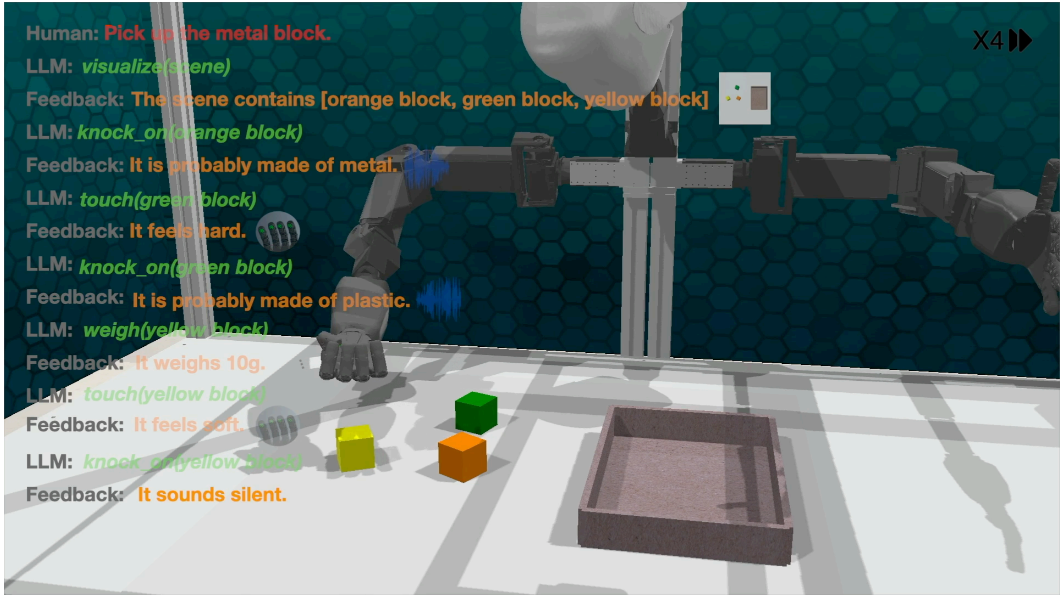
Human: Pick up the metal block.
LLM: *visualize(scene)*
Feedback: The scene contains [orange block, green block, yellow block]
LLM: *knock_on(orange block)*
Feedback: It is probably made of metal.
LLM: *touch(green block)*
Feedback: It feels hard.
LLM: *knock_on(green block)*
Feedback: It is probably made of plastic.
LLM: *weigh(yellow block)*
Feedback: It weighs 10g.
LLM: *touch(yellow block)*
Feedback: It feels soft.
LLM: *knock_on(yellow block)*
Feedback: It sounds silent.

NICOL Robot in WTM

Chat with the environment: Interactive multimodal perception using large language models

# High-Level Planning: Matcha

- Task
  - Find an object with a certain property.
- Example
  - Pick up the metal block.
- Use the sound, touch, and weight of the objects.
- LLM guides the search and infers the right object based on the clues.



Human: Pick up the metal block.

LLM: *visualize(scene)*

Feedback: The scene contains [orange block, green block, yellow block]

LLM: *knock_on(orange block)*

Feedback: It is probably made of metal.

LLM: *touch(green block)*

Feedback: It feels hard.

LLM: *knock_on(green block)*

Feedback: It is probably made of plastic.

LLM: *weigh(yellow block)*

Feedback: It weighs 10g.

LLM: *touch(yellow block)*

Feedback: It feels soft.

LLM: *knock_on(yellow block)*

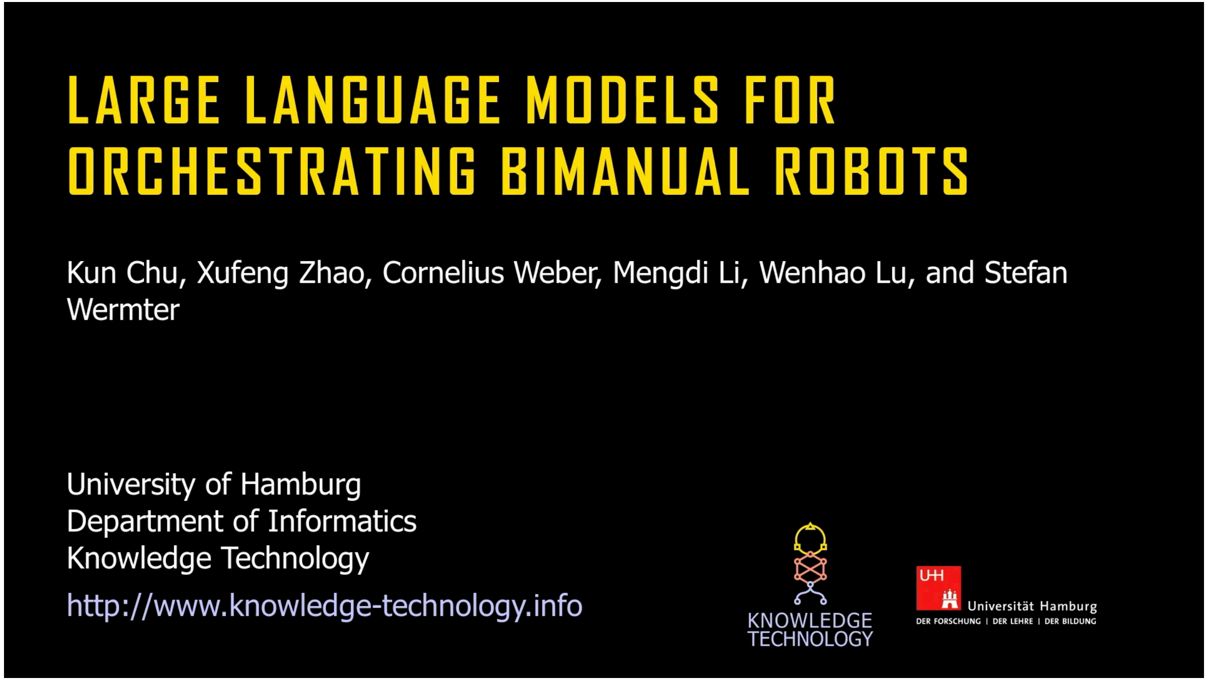
Feedback: It sounds silent.

NICOL Robot in WTM

Chat with the environment: Interactive multimodal perception using large language models

# Low-Level Control: Labor

- LLM guides **bimanual manipulation**



LARGE LANGUAGE MODELS FOR ORCHESTRATING BIMANUAL ROBOTS

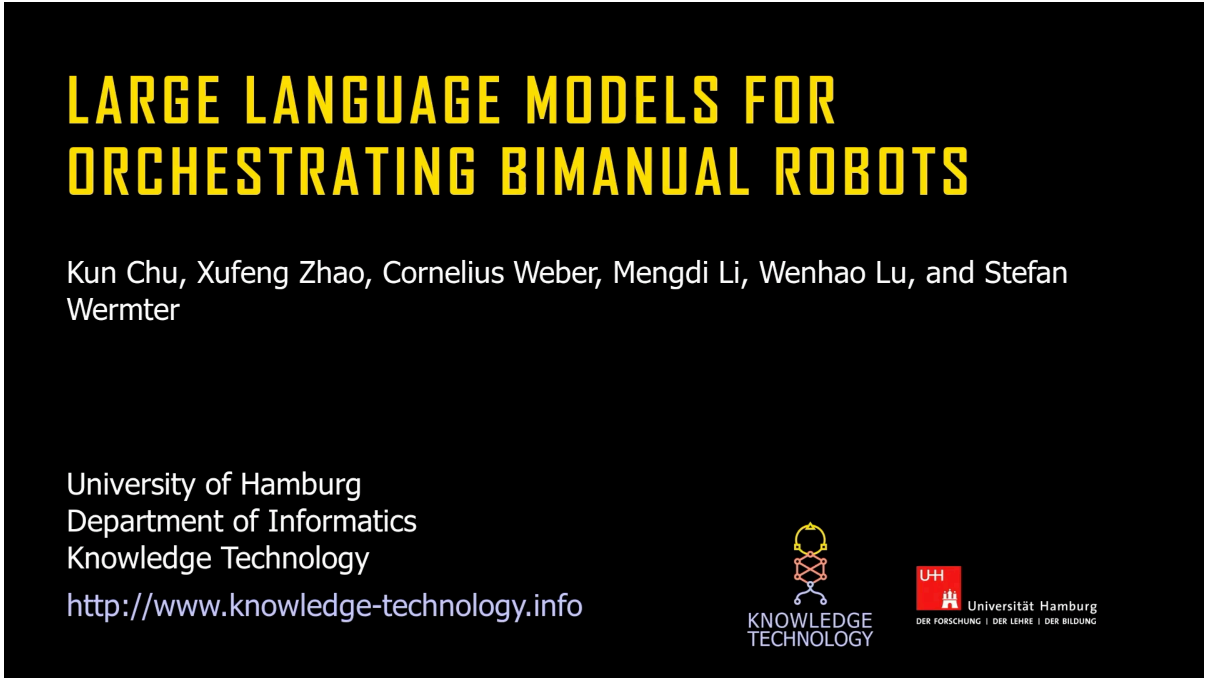
Kun Chu, Xufeng Zhao, Cornelius Weber, Mengdi Li, Wenhao Lu, and Stefan Wermter

University of Hamburg
Department of Informatics
Knowledge Technology
http://www.knowledge-technology.info

KNOWLEDGE TECHNOLOGY
UH
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

# Low-Level Control: Labor

- LLM guides **bimanual manipulation**



LARGE LANGUAGE MODELS FOR ORCHESTRATING BIMANUAL ROBOTS

Kun Chu, Xufeng Zhao, Cornelius Weber, Mengdi Li, Wenhao Lu, and Stefan Wermter

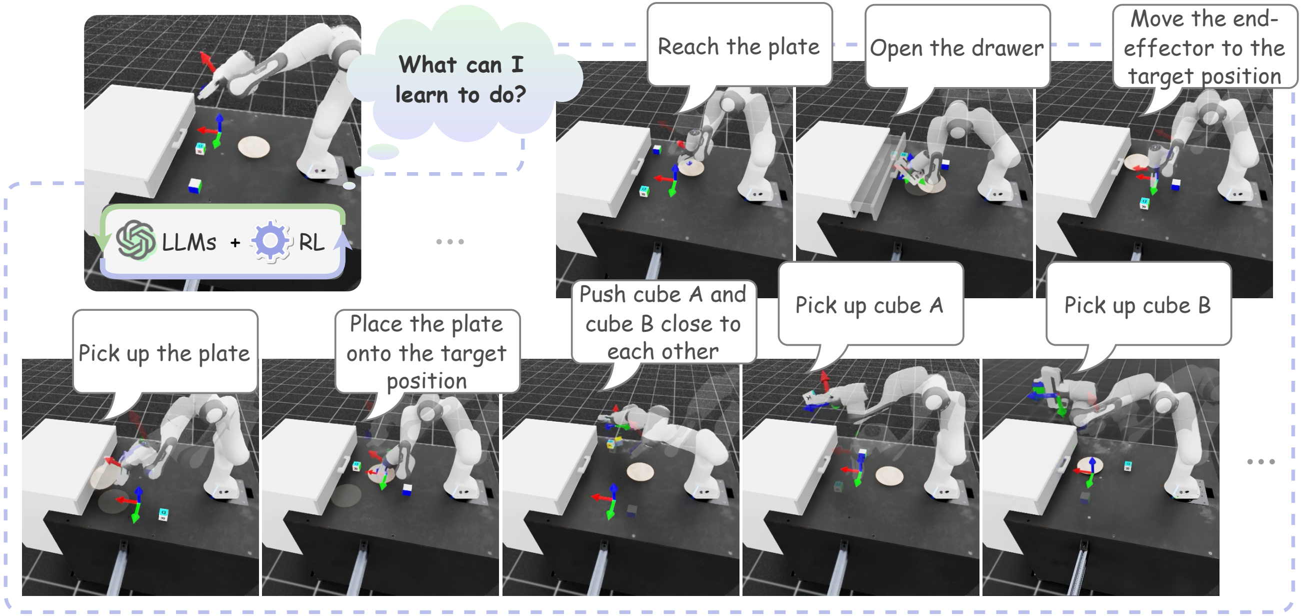
University of Hamburg
Department of Informatics
Knowledge Technology
http://www.knowledge-technology.info

KNOWLEDGE TECHNOLOGY
UHH
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Large Language Models for Orchestrating Bimanual Robots

# Low-Level Control: Agentic Skill Discovery

- LLM **proposes, learns, collects** skills for robot


What can I learn to do?
LLMs + RL
...
Reach the plate
Open the drawer
Move the end-effector to the target position
Pick up the plate
Place the plate onto the target position
Push cube A and cube B close to each other
Pick up cube A
Pick up cube B
...

Agentic Skill Discovery

# Summary

- An LLM is a transformer decoder and training an LLM is done in several stages.
- The performance of an LLM can be improved with diverse methods.
- LLMs can be (partially) interpreted.
- LLMs can be used both for high-level planning and low-level control in robotics.

# Resources

- A Survey of Large Language Models
- Ahead of AI
- Build a Large Language Model (From Scratch)
- Transformer Interpretability Tutorial