



# Large Language Model

Kasidis Kanwatchara



# Outline

## Lecture

1. What is LLM?
2. Instruction-tuned LLM
3. Prompt engineering
4. RAG
5. Agentic LLM

## Demo

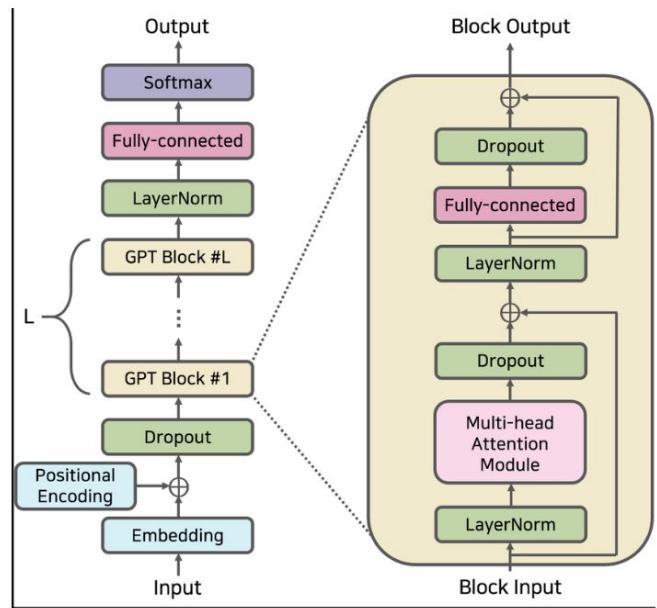
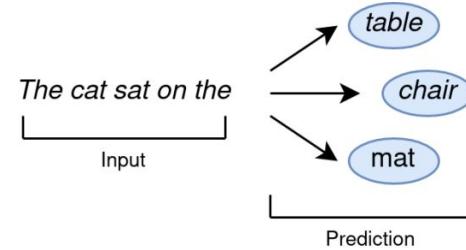
1. Prompt engineering
2. RAG with LangChain
3. Agentic RAG with LangChain and LangGraph
  - a. Each RAG task is defined by LangChain
  - b. Agentic workflow is created by LangGraph

# Large Language Model (LLM)

# Large Language Model (LLM)

A (very) large model that predicts the next word from an input sentence.

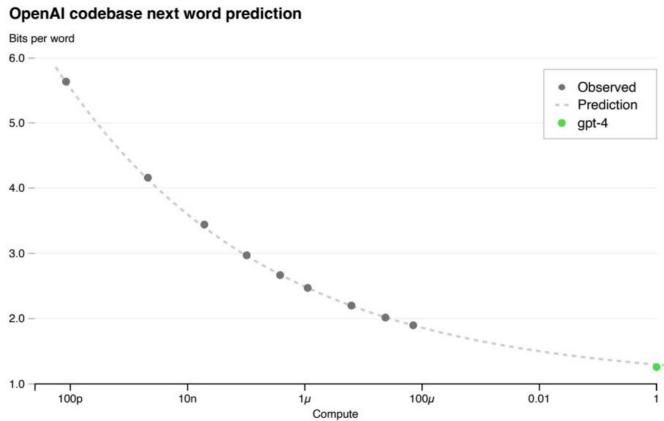
Just stacks of transformer layers trained on a giant pile of text.



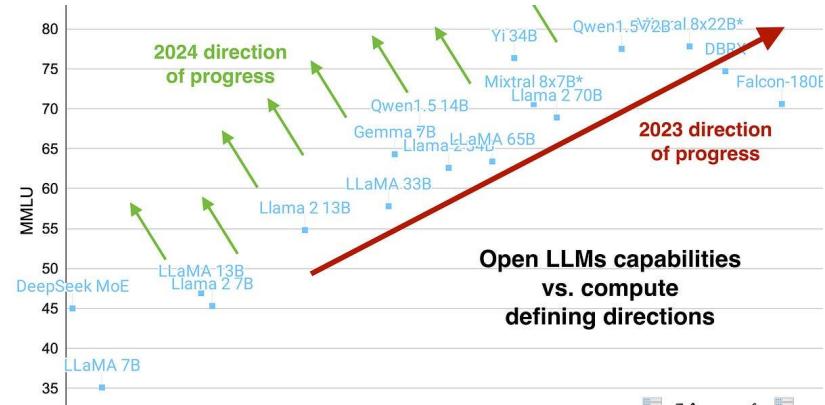
Conceptual architecture of a GPT model.

# Large Language Model (LLM)

More data + more compute + bigger model = better model



**Figure 1.** Performance of GPT-4 and smaller models. The metric is final loss on a dataset derived from our internal codebase. This is a convenient, large dataset of code tokens which is not contained in the training set. We chose to look at loss because it tends to be less noisy than other measures across different amounts of training compute. A power law fit to the smaller models (excluding GPT-4) is shown as the dotted line; this fit accurately predicts GPT-4's final loss. The x-axis is training compute normalized so that GPT-4 is 1.



Louie Peters @\_LouiePeters

LLaMA 3.1 405B likely cost ~\$60m after training for ~100 days on a 16k H100 datacentre (maybe ~\$850m capex). This is a huge free gift to open source AI & anyone building with AI!

It is not yet however a next generation LLM (eg GPT-5, Claude Opus 3.5, Gemini Ultra 1.5/2.0, Grok-3, Llma-4) which i expect will cost

# In-context Learning (ICL)

ICL is an emergent ability of LLMs popularized by the GPT3 paper.

LLMs can “learn” new tasks from inputs (prompts) without training.

The three settings we explore for in-context learning

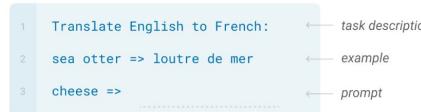
## Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



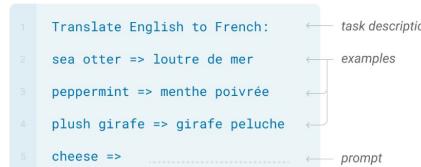
## One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



## Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



Traditional fine-tuning (not used for GPT-3)

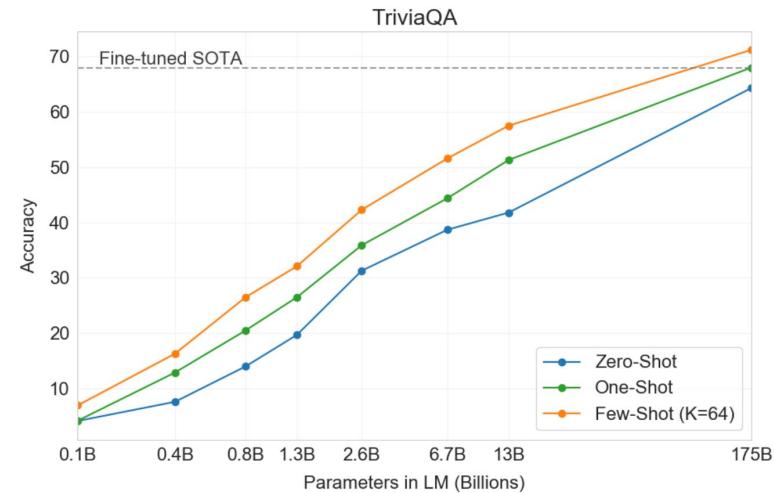
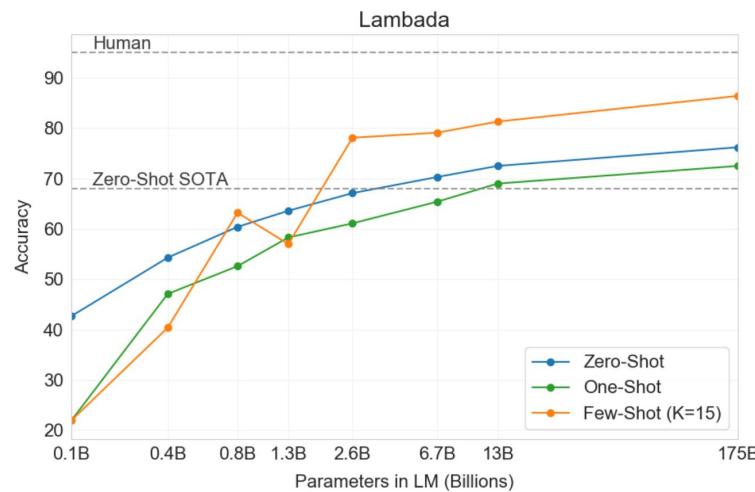
## Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



# In-context learning (ICL)

ICL allows us to use LLMs on any natural language task without having to fine-tune the models.



# In-context learning (ICL)

## Some examples

Input: 2014-06-01  
Output: !06!01!2014!  
Input: 2007-12-13  
Output: !12!13!2007!  
Input: 2010-09-23  
Output: !09!23!2010!  
Input: 2005-07-23  
Output: !07!23!2005!

*in-context  
examples*

*test example*

! --- *model completion*

### Demonstrations

- Circulation revenue has increased by 5% in Finland. \n Positive  
Panostaja did not disclose the purchase price. \n Neutral  
Paying off the national debt will be extremely painful. \n Negative  
The acquisition will have an immediate positive impact. \n \_\_\_\_\_

*Test input*



# In-context learning (ICL)

Still no consensus on how or why ICL works.

One hypothesis is that LLMs learn patterns from a few examples in the input prompt. The model makes predictions by recognizing similarities and structures in the provided context.

Work	Key Words	Models	Tasks
<b>Theoretical Perspective</b>			
(Elhage et al., 2021)	Mechanistic Interpretability	Transformer <sup>†</sup>	-
(Olsson et al., 2022)	Mechanistic Interpretability	Transformer	-
(Edelman et al., 2024)	Mechanistic Interpretability	Transformer <sup>†</sup>	Markov Chain modeling
(Swaminathan et al., 2023)	Mechanistic Interpretability	Transformer	Next token prediction
(Todd et al., 2024)	Mechanistic Interpretability	Llama 2, GPT <sup>‡</sup>	Antonym, etc.
(Bai et al., 2023)	Mechanistic Interpretability	Transformer <sup>†</sup>	Regression
(Garg et al., 2022)	Regression Function Learning	Transformer <sup>†</sup>	Regression
(Li et al., 2023b)	Regression Function Learning	Transformer <sup>†</sup>	Regression
(Li et al., 2023a)	Regression Function Learning	Transformer <sup>†</sup>	Regression
(Akyürek et al., 2023)	Regression Function Learning	Transformer <sup>†</sup>	Regression
(Guo et al., 2024)	Regression Function Learning	GPT <sup>‡</sup>	Representation Regression
(Dai et al., 2023)	Gradient Descent, Meta-Optimization	GPT <sup>‡</sup>	Classification
(von Oswald et al., 2023a)	Gradient Descent, Meta-Optimization	Transformer <sup>†</sup>	Regression
(von Oswald et al., 2023b)	Gradient Descent, Meta-Optimization	Transformer <sup>†</sup>	Regression
(Deutch et al., 2024)	Gradient Descent, Meta-Optimization	Transformer <sup>†</sup>	Classification
(Shen et al., 2024)	Gradient Descent, Meta-Optimization	LLaMa, GPT <sup>‡</sup>	Classification
(Fu et al., 2024)	Gradient Descent, Meta-Optimization	Transformer, LSTM	Regression
(Xie et al., 2022)	Bayesian Inference	Transformer <sup>†</sup> , LSTM	Next token prediction
(Wang et al., 2023b)	Bayesian Inference	GPT <sup>‡</sup>	Classification
(Wies et al., 2023)	Bayesian Inference	-	-
(Jiang, 2023)	Bayesian Inference	GPT <sup>‡</sup>	Synthetic Generation
(Zhang et al., 2023b)	Bayesian Inference	Transformer <sup>†</sup>	-
(Panwar et al., 2024)	Bayesian Inference	Transformer	Regression
(Jeon et al., 2024)	Bayesian Inference	Transformer <sup>†</sup>	Regression
(Bigelow et al., 2024)	Bayesian Inference	GPT <sup>‡</sup>	Sequence generation
<b>Empirical Perspective</b>			
(Shin et al., 2022)	DATA Domain	GPT-3	Classification, etc.
(Han et al., 2023)	DATA Domain, DATA Distribution	OPT <sup>‡</sup>	Classification
(Raventós et al., 2023)	Task Diversity	GPT-2	Regression
(Razeghi et al., 2022)	DATA Term frequency	GPT <sup>‡</sup>	Reasoning
(Kandpal et al., 2023)	DATA Term frequency	GPT-3 <sup>‡</sup>	Question Answering
(Chan et al., 2022)	DATA Distribution	Transformer	Classification
(Yadlowsky et al., 2023)	DATA Distribution	GPT-2 <sup>‡</sup>	Regression
(Hendel et al., 2023)	Task Diversity	LLaMA	Translation, etc.
(Wei et al., 2022b)	Model Scale	GPT-3 <sup>‡</sup>	Classification
(Schaeffer et al., 2023)	Model Scale, Evaluation Metric	GPT-3 <sup>‡</sup>	Classification
(Tay et al., 2023)	Pre-training Objective	UL2 <sup>‡</sup>	Classification, etc.
(Kirsch et al., 2024)	Model Architecture	Transformer <sup>†</sup>	Classification
(Singh et al., 2023)	Model Architecture	LLaMA <sup>†</sup>	Synthetic generation
(Yousefi et al., 2024)	Embeddings	Llama, Vicuna	Regression
(Akyürek et al., 2024)	Model Architecture	Transformer, LSTM, etc.	Language learning
(Lu et al., 2022)	Demonstration Order	GPT <sup>‡</sup>	Classification
(Liu et al., 2024)	Demonstration Order	GPT-3.5 <sup>‡</sup>	Question Answering
(Zhao et al., 2021)	Demonstration	GPT <sup>‡</sup>	Classification, IE, IR
(Liu et al., 2022)	Demonstration Order	GPT-3	Classification, QA, etc.
(Min et al., 2022)	Input-Label Mapping	GPT <sup>‡</sup>	Classification, etc.
(Yoo et al., 2022)	Input-Label Mapping	GPT <sup>‡</sup>	Classification
(Wei et al., 2023)	Input-Label Mapping	GPT-3 <sup>‡</sup>	Classification, QA, etc.
(Pan et al., 2023)	Input-Label Mapping	GPT <sup>‡</sup>	Classification, etc.
(Lin and Lee, 2024)	Input-Label Mapping	LLama, LSTM, etc.	Classification, etc.
(Kossen et al., 2024)	Input-Label Mapping	LLaMA <sup>‡</sup>	Classification
(Fang et al., 2023)	Input-Label Mapping	GPT <sup>‡</sup>	Classification
(Si et al., 2023)	Input-Label Mapping	GPT-3 <sup>‡</sup>	Classification, QA, etc.
(Wang et al., 2023a)	Input-Label Mapping	GPT2-XL	Classification

# Objective mismatch

Language models are trained on the next word prediction objective.

Not suitable for instruction following:

- Might misinterpret the request or require very specific phrasing to work well

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.

# Objective mismatch

What we want is LLMs that follow instructions better.

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.

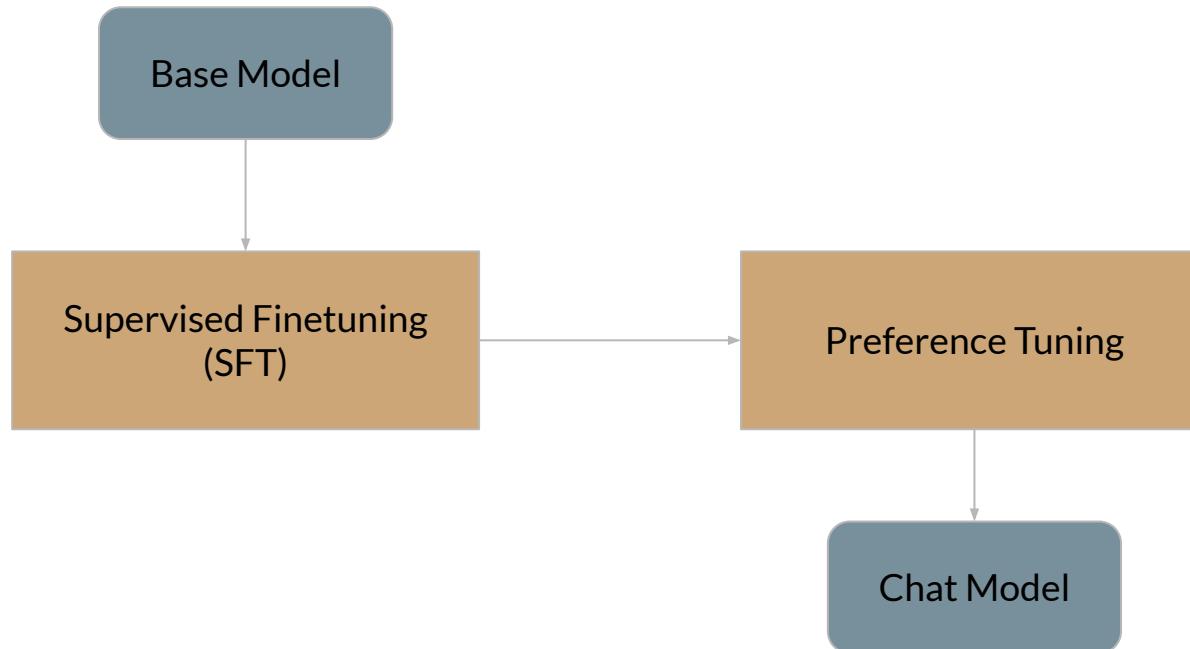
Explain evolution to a 6 year old.

InstructGPT

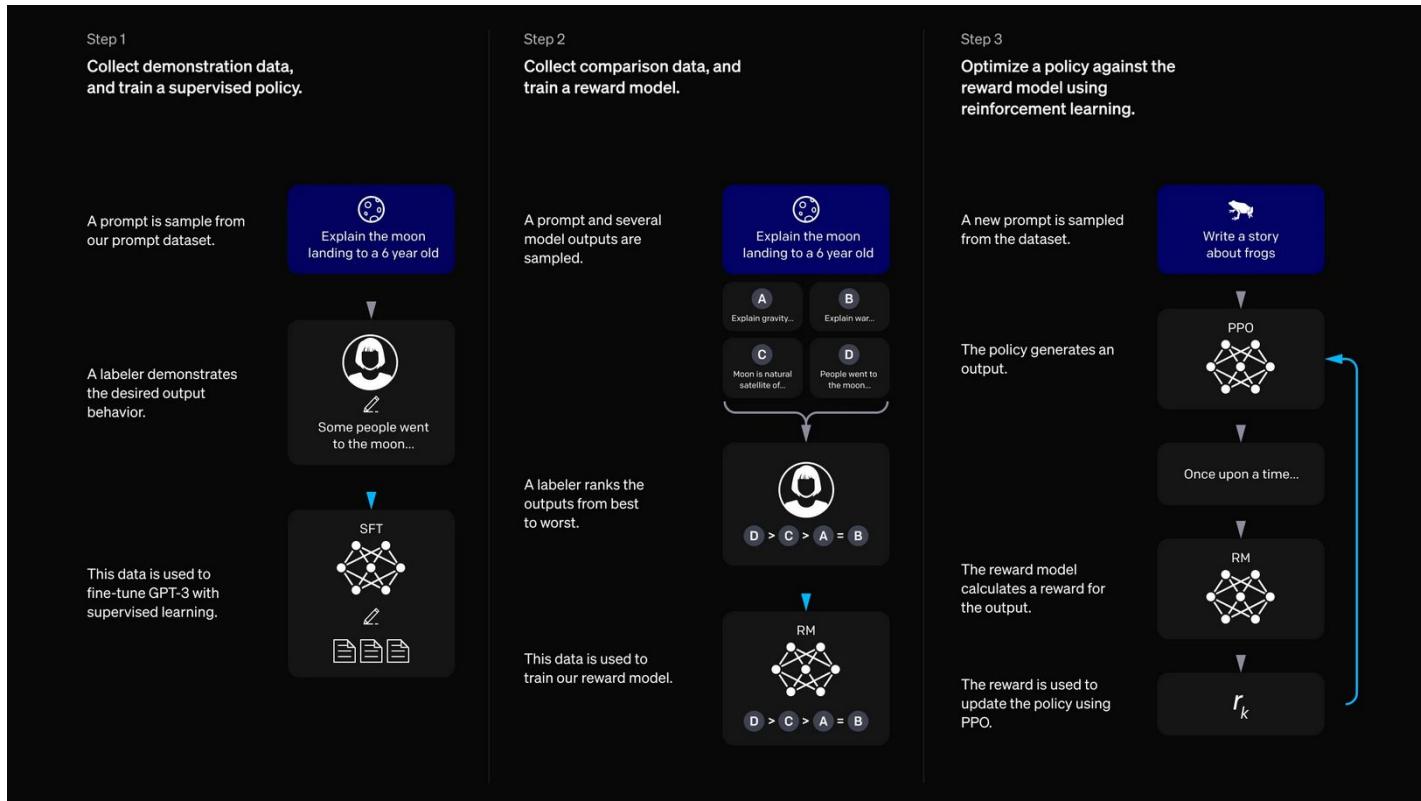
People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.

# Instruction-tuned LLM

# Instruction Tuning (Post-training)



# Reinforcement with Human Feedback (RLHF)



# Direct Preference Optimization (DPO)

Just doing a simple classification is enough. No RL needed.

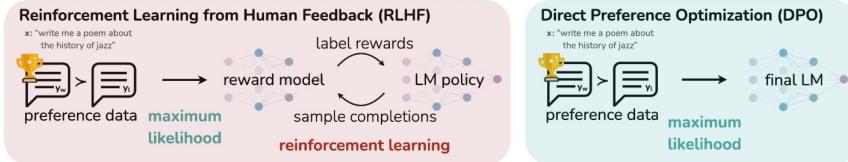


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an *implicit* reward model whose corresponding optimal policy can be extracted in closed form.

Here are our findings:

1. **PPO Gets Similar Average Scores with DPO in this Non-Tuned Setup** Overall, we found that PPO could reach a comparable level of performance to DPO (albeit slightly lower) in this controlled setup.
2. **PPO is More Computationally Expensive** The PPO runtime is roughly 28 hours using two nodes, whereas the DPO runtime is about 4 hours using a single node.

Algorithm	LR	$\gamma - \beta$ ratio	$\beta$	Epochs	Batch Size	Average Score
SFT Base	-	-	-	-	-	55.7
SimPO	5.00E-07	0.5	2	1	128	51.8
SimPO	5.00E-07	0.3	10	1	128	52.9
DPO	5.00E-07	-	0.1	3	32	55.2
PPO	1.00E-06	-	0.0325	1	64	54.5
PPO	1.00E-06	-	0.05	1	64	55.5
DPO-norm	1.00E-07	-	5	3	32	56.1
DPO-norm	5.00E-07	-	10	3	32	55.2
DPO-norm	5.00E-07	-	15	3	32	55.7
DPO-norm	5.00E-07	-	2	3	32	46.8
DPO-norm	5.00E-07	-	5	3	32	53.4
DPO-norm	5.00E-07	-	5	1	32	<b>57.3</b>

**Table 18** Hyperparameters and algorithms examined for DPO tuning. We use UltraFeedback as the training dataset in all cases, and train on top of an early T<sub>0</sub>LU 3 version. DPO-norm refers to the length-normalized DPO variant proposed in Meng et al. (2024). We explore hyperparameters suggested by prior work (Meng et al., 2024; Ivison et al., 2023). For PPO, we train reward models on UltraFeedback and reuse prompts during online training, following the hyperparameters in Ivison et al. (2024). We find that length-normalized DPO performs best overall.

# Group Relative Policy Optimization (GRPO)

PPO is an actor-critic RL algorithm -> it needs to train two models.

But GRPO needs just one.

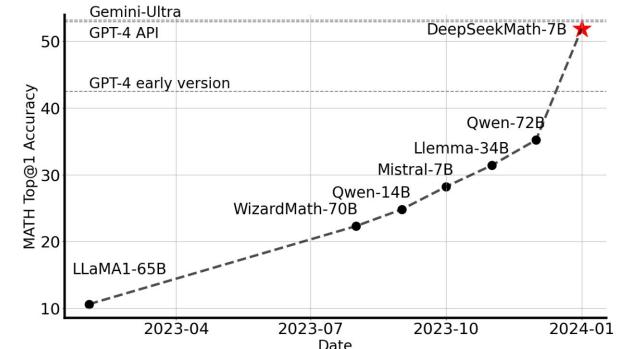
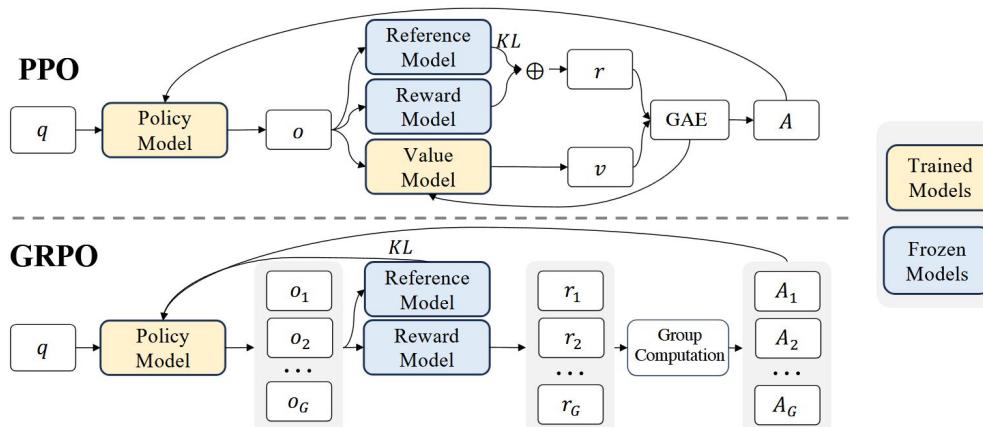


Figure 1 | Top1 accuracy of open-source models on the competition-level MATH benchmark (Hendrycks et al., 2021) without the use of external toolkits and voting techniques.

# Prompt Engineering

# Prompting Principles

Principle 1: Write clear and specific instructions

Principle 2: Give the model time to “think”

# Tactic 1.1: Delimiters to indicate parts of the input

Task: Summarize the text

- Delimiters can be anything like: ``, `""`, <>, <tag> </tag>, :

```
In [3]: text = f"""
You should express what you want a model to do by \
providing instructions that are as clear and \
specific as you can possibly make them. \
This will guide the model towards the desired output, \
and reduce the chances of receiving irrelevant \
or incorrect responses. Don't confuse writing a \
clear prompt with writing a short prompt. \
In many cases, longer prompts provide more clarity \
and context for the model, which can lead to \
more detailed and relevant outputs.
"""

prompt = f"""
Summarize the text delimited by triple backticks \
into a single sentence.
``{text}``
"""

response = get_completion(prompt)
print(response)
```

Providing clear and specific instructions to a model is essential for guiding it towards the desired output and reducing the chances of irrelevant or incorrect responses, with longer prompts often providing more clarity and context for more detailed and relevant outputs.

# Tactic 1.2: Structured Output

Task: Generate structured outputs

```
In [4]: prompt = f"""
Generate a list of three made-up book titles along \
with their authors and genres.
Provide them in JSON format with the following keys:
book_id, title, author, genre.
"""

response = get_completion(prompt)
print(response)
```

```
[{"book_id": 1, "title": "The Midnight Garden", "author": "Elena Rivers", "genre": "Fantasy"}, {"book_id": 2, "title": "Echoes of the Past", "author": "Nathan Black", "genre": "Mystery"}, {"book_id": 3, "title": "Whispers in the Wind", "author": "Samantha Reed", "genre": "Romance"}]
```

# Tactic 1.3: Set Base Case

Task: Convert instructions to line-by-line steps

```
In [5]: text_1 = f"""
Making a cup of tea is easy! First, you need to get some
water boiling. While that's happening, \
grab a cup and put a tea bag in it. Once the water is \
hot enough, just pour it over the tea bag. \
Let it sit for a bit so the tea can steep. After a \
few minutes, take out the tea bag. If you \
like, you can add some sugar or milk to taste. \
And that's it! You've got yourself a delicious \
cup of tea to enjoy.
"""

prompt = f"""
You will be provided with text delimited by triple quotes.
If it contains a sequence of instructions, \
re-write those instructions in the following format:
```

```
Step 1 - ...
Step 2 - ...
...
Step N - ...
```

```
If the text does not contain a sequence of instructions, \
then simply write "No steps provided."
```

```
\""""\n{text_1}\n\""""
response = get_completion(prompt)
print("Completion for Text 1:")
print(response)
```

```
Completion for Text 1:
Step 1 - Get some water boiling.
Step 2 - Grab a cup and put a tea bag in it.
Step 3 - Pour the hot water over the tea bag.
Step 4 - Let the tea steep for a few minutes.
Step 5 - Remove the tea bag.
Step 6 - Add sugar or milk to taste.
Step 7 - Enjoy your delicious cup of tea.
```

```
In [6]: text_2 = f"""
The sun is shining brightly today, and the birds are \
singing. It's a beautiful day to go for a \
walk in the park. The flowers are blooming, and the \
trees are swaying gently in the breeze. People \
are out and about, enjoying the lovely weather. \
Some are having picnics, while others are playing \
games or simply relaxing on the grass. It's a \
perfect day to spend time outdoors and appreciate the \
beauty of nature.
"""

prompt = f"""
You will be provided with text delimited by triple quotes.
If it contains a sequence of instructions, \
re-write those instructions in the following format:
```

```
Step 1 - ...
Step 2 - ...
...
Step N - ...
```

```
If the text does not contain a sequence of instructions, \
then simply write "No steps provided."
```

```
\""""\n{text_2}\n\""""
response = get_completion(prompt)
print("Completion for Text 2:")
print(response)
```

```
Completion for Text 2:
No steps provided.
```

# Tactic 1.4: Provide Examples

Task: Answer in a consistent style

```
In [7]: prompt = f"""
Your task is to answer in a consistent style.

<child>: Teach me about patience.

<grandparent>: The river that carves the deepest \
valley flows from a modest spring; the \
grandest symphony originates from a single note; \
the most intricate tapestry begins with a solitary thread.

<child>: Teach me about resilience.
"""

response = get_completion(prompt)
print(response)
```

```
<grandparent>: The tallest trees weather the strongest storms; the brightest stars shine in the darkest nights; the strongest hearts are forged in the hottest fires.
```

# Prompting Principles

Principle 1: Write clear and specific instructions

Principle 2: Give the model time to “think”

# Tactic 2.1: Lay out the steps to complete the task

```
In [8]: text = """  
In a charming village, siblings Jack and Jill set out on \  
a quest to fetch water from a hilltop \  
well. As they climbed, singing joyfully, misfortune \  
struck-Jack tripped on a stone and tumbled \  
down the hill, with Jill following suit. \  
Though slightly battered, the pair returned home to \  
comforting embraces. Despite the mishap, \  
their adventurous spirits remained undimmed, and they \  
continued exploring with delight.  
"""
```

```
# example 1  
prompt_1 = """  
Perform the following actions:  
1 - Summarize the following text delimited by triple \  
backticks with 1 sentence.  
2 - Translate the summary into French.  
3 - List each name in the French summary.  
4 - Output a json object that contains the following \  
keys: french_summary, num_names.
```

```
Separate your answers with line breaks.
```

```
Text:  
```{text}```  
"""  
  
response = get_completion(prompt_1)  
print("Completion for prompt 1:")  
print(response)
```

Completion for prompt 1:

1 - Jack and Jill, siblings from a charming village, go on a quest to fet  
ch water from a hilltop well, but encounter misfortune along the way.

2 - Jack et Jill, frère et sœur d'un charmant village, partent en quête d  
'eau d'un puits au sommet d'une colline, mais rencontrent des malheurs en  
chemin.

3 - Jack, Jill

4 -  
{  
 "french\_summary": "Jack et Jill, frère et sœur d'un charmant village, p  
artent en quête d'eau d'un puits au sommet d'une colline, mais rencontren  
t des malheurs en chemin.",  
 "num\_names": 2  
}

# Tactic 2.2: Let the model work out its own solution first

## Without “thinking”

```
In [10]: prompt = f"""
Determine if the student's solution is correct or not.

Question:
I'm building a solar power installation and I need \
help working out the financials.
- Land costs $100 / square foot
- I can buy solar panels for $250 / square foot
- I negotiated a contract for maintenance that will cost \
me a flat $100k per year, and an additional $10 / square \
foot
What is the total cost for the first year of operations \
as a function of the number of square feet.

Student's Solution:
Let x be the size of the installation in square feet.
Costs:
1. Land cost: 100x
2. Solar panel cost: 250x
3. Maintenance cost: 100,000 + 100x
Total cost: 100x + 250x + 100,000 + 100x = 450x + 100,000
"""
response = get_completion(prompt)
print(response)
```

The student's solution is correct. The total cost for the first year of operations as a function of the number of square feet is indeed  $450x + 100,000$ .

## With “thinking”

Note that the student's solution is actually not correct.

We can fix this by instructing the model to work out its own solution first.

```
In [11]: prompt = f"""
Your task is to determine if the student's solution \
is correct or not.
To solve the problem do the following:
- First, work out your own solution to the problem including the final \
total cost.
- Then compare your solution to the student's solution \
and evaluate if the student's solution is correct or not.
Don't decide if the student's solution is correct until \
you have done the problem yourself.

Use the following format:
Question:
```
question here
```
Student's solution:
```
student's solution here
```
Actual solution:
```
steps to work out the solution and your solution here
```
Is the student's solution the same as actual solution \
just calculated?
```
yes or no
```
Student grade:
```
correct or incorrect
```
"""

response = get_completion(prompt)
print(response)
```

```
Question:
```
I'm building a solar power installation and I need help \
working out the financials.
- Land costs $100 / square foot
- I can buy solar panels for $250 / square foot
- I negotiated a contract for maintenance that will cost \
me a flat $100k per year, and an additional $10 / square \
foot
What is the total cost for the first year of operations \
as a function of the number of square feet.
```
Student's solution:
```
Let x be the size of the installation in square feet.
Costs:
1. Land cost: 100x
2. Solar panel cost: 250x
3. Maintenance cost: 100,000 + 100x
Total cost: 100x + 250x + 100,000 + 100x = 450x + 100,000
```
Actual solution:
```
response = get_completion(prompt)
print(response)
```
Let x be the size of the installation in square feet.
```

```
Costs:
1. Land cost: 100x
2. Solar panel cost: 250x
3. Maintenance cost: 100,000 + 10x

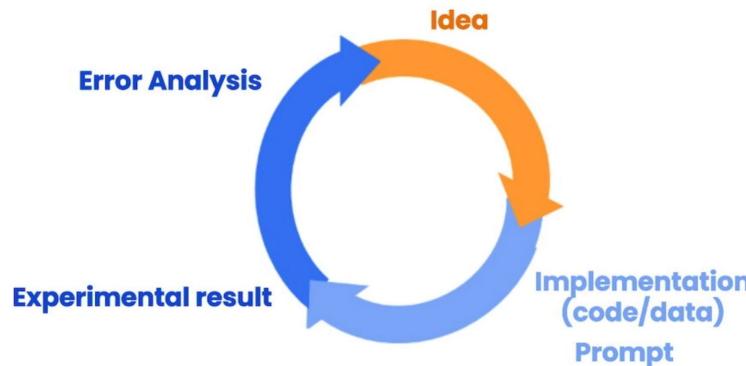
Total cost: 100x + 250x + 100,000 + 10x = 360x + 100,000

Is the student's solution the same as actual solution just \
calculated:
```
No
```
Student grade:
```
Incorrect
```

```

# Iterative Prompt Development

There is no “the perfect prompts” that works with every application. So it is very important to learn how to develop a prompt that works for your specific application.



## Prompt guidelines

- Be clear and specific
- Analyze why result does not give desired output.
- Refine the idea and the prompt
- Repeat

# Example

## The Task

Generate a marketing product description from a product fact sheet

```
In [3]: fact_sheet_chair = """
OVERVIEW
- Part of a beautiful family of mid-century inspired office furniture, including filing cabinets, desks, bookcases, meeting tables, and more.
- Several options of shell color and base finishes.
- Available with plastic back and front upholstery (SWC-100) or full upholstery (SWC-110) in 10 fabric and 6 leather options.
- Base finish options are: stainless steel, matte black, gloss white, or chrome.
- Chair is available with or without armrests.
- Suitable for home or business settings.
- Qualified for contract use.

CONSTRUCTION
- 5-wheel plastic coated aluminum base.
- Pneumatic chair adjust for easy raise/lower action.

DIMENSIONS
- WIDTH 53 CM | 20.87"
- DEPTH 51 CM | 20.08"
- HEIGHT 80 CM | 31.50"
- SEAT HEIGHT 44 CM | 17.32"
- SEAT DEPTH 41 CM | 16.14"

OPTIONS
- Soft or hard-floor caster options.
- Two choices of seat foam densities: medium (1.8 lb/ft³) or high (2.8 lb/ft³)
- Armless or 8 position PU armrests

MATERIALS
SHELL BASE GLIDER
- Cast Aluminum with modified nylon PA6/PA66 coating.
- Shell thickness: 10 mm.
SEAT
- HD36 foam

COUNTRY OF ORIGIN
- Italy
"""

fact_sheet_chair
```

## The Prompt

```
In [4]: prompt = f"""
Your task is to help a marketing team create a
description for a retail website of a product based
on a technical fact sheet.
```

```
Write a product description based on the information
provided in the technical specifications delimited by
triple backticks.
```

```
Technical specifications: ```{fact_sheet_chair}```
"""
response = get_completion(prompt)
print(response)
```

Introducing our stylish and versatile mid-century inspired office chair, perfect for both home and business settings. This chair is part of a beautiful family of office furniture that includes filing cabinets, desks, bookcases, meeting tables, and more.

Customize your chair with several options of shell color and base finishes to suit your personal style. Choose between plastic back and front upholstery or full upholstery in a variety of fabric and leather options. The base finish options include stainless steel, matte black, gloss white, or chrome. You can also choose to have armrests or go for a sleek armless design.

Constructed with a 5-wheel plastic coated aluminum base, this chair features a pneumatic chair adjust for easy raise/lower action. The dimensions of the chair are as follows: width 53 cm, depth 51 cm, height 80 cm, seat height 44 cm, and seat depth 41 cm.

Personalize your chair even further with options such as soft or hard-floor or caster options, two choices of seat foam densities (medium or high), and the choice between armless or 8 position PU armrests.

Made with high-quality materials, including a cast aluminum shell with modified nylon coating and HD36 foam seat, this chair is designed for durability and comfort. Plus, it is qualified for contract use, making it a reliable choice for any workspace.

Add a touch of Italian elegance to your office with this stylish and functional office chair. Elevate your workspace with our mid-century inspired chair that combines style, comfort, and functionality seamlessly.

Too long!

# Example

## The Task: make it shorter

Generate a marketing product description from a product fact sheet

```
In [3]: fact_sheet_chair = """
OVERVIEW
- Part of a beautiful family of mid-century inspired office furniture, including filing cabinets, desks, bookcases, meeting tables, and more.
- Several options of shell color and base finishes.
- Available with plastic back and front upholstery (SWC-100) or full upholstery (SWC-110) in 10 fabric and 6 leather options.
- Base finish options are: stainless steel, matte black, gloss white, or chrome.
- Chair is available with or without armrests.
- Suitable for home or business settings.
- Qualified for contract use.

CONSTRUCTION
- 5-wheel plastic coated aluminum base.
- Pneumatic chair adjust for easy raise/lower action.

DIMENSIONS
- WIDTH 53 CM | 20.87"
- DEPTH 51 CM | 20.08"
- HEIGHT 80 CM | 31.50"
- SEAT HEIGHT 44 CM | 17.32"
- SEAT DEPTH 41 CM | 16.14"

OPTIONS
- Soft or hard-floor caster options.
- Two choices of seat foam densities: medium (1.8 lb/ft3) or high (2.8 lb/ft3)
- Armless or 8 position PU armrests

MATERIALS
SHELL BASE GLIDER
- Cast Aluminum with modified nylon PA6/PA66 coating.
- Shell thickness: 10 mm.

SEAT
- HD36 foam

COUNTRY OF ORIGIN
- Italy
"""
```

## The Prompt

- Limit the number of words/sentences/characters.

```
In [5]: prompt = f"""
Your task is to help a marketing team create a description for a retail website of a product based on a technical fact sheet.
```

Write a product description based on the information provided in the technical specifications delimited by triple backticks.

Use at most 50 words.

```
Technical specifications: ```{fact_sheet_chair}```
"""
response = get_completion(prompt)
print(response)
```

Introducing our versatile and stylish mid-century office chair, available in a range of colors and finishes. With adjustable height and comfortable upholstery options, this chair is perfect for both home and business use. Made in Italy with quality materials, it's a perfect blend of form and function.

Too general!

# Example

## The Task: make it more technical

Generate a marketing product description from a product fact sheet

```
In [3]: fact_sheet_chair = """
OVERVIEW
- Part of a beautiful family of mid-century inspired office furniture, including filing cabinets, desks, bookcases, meeting tables, and more.
- Several options of shell color and base finishes.
- Available with plastic back and front upholstery (SWC-100) or full upholstery (SWC-118) in 10 fabric and 6 leather options.
- Base finish options are: stainless steel, matte black, gloss white, or chrome.
- Chair is available with or without armrests.
- Suitable for home or business settings.
- Qualified for contract use.

CONSTRUCTION
- 5-wheel plastic coated aluminum base.
- Pneumatic chair adjust for easy raise/lower action.

DIMENSIONS
- WIDTH 53 CM | 20.87"
- DEPTH 51 CM | 20.08"
- HEIGHT 80 CM | 31.50"
- SEAT HEIGHT 44 CM | 17.32"
- SEAT DEPTH 41 CM | 16.14"

OPTIONS
- Soft or hard-floor caster options.
- Two choices of seat foam densities: medium (1.8 lb/ft3) or high (2.8 lb/ft3)
- Armless or 8 position PU armrests

MATERIALS
SHELL BASE GLIDER
- Cast Aluminum with modified nylon PA6/PA66 coating.
- Shell thickness: 10 mm.
SEAT
- HD36 foam

COUNTRY OF ORIGIN
- Italy
"""

fact_sheet_chair
```

## The Prompt

```
In [8]: prompt = f"""
Your task is to help a marketing team create a description for a retail website of a product based on a technical fact sheet.
```

Write a product description based on the information provided in the technical specifications delimited by triple backticks.

The description is intended for furniture retailers, so should be technical in nature and focus on the materials the product is constructed from.

At the end of the description, include every 7-character Product ID in the technical specification.

Use at most 50 words.

```
Technical specifications: ```{fact_sheet_chair}```
"""
response = get_completion(prompt)
print(response)
```

Introducing our versatile and stylish office chair, part of a mid-century inspired furniture collection. Choose from a variety of shell colors and base finishes to suit your space. Constructed with a durable aluminum base and high-density foam seat for comfort. Perfect for home or business use.

Too general!

# Example

## The Task: format it

Generate a marketing product description from a product fact sheet

```
In [3]: fact_sheet_chair = """
OVERVIEW
- Part of a beautiful family of mid-century inspired office furniture,
  including filing cabinets, desks, bookcases, meeting tables, and more.
- Several options of shell color and base finishes.
- Available with plastic back and front upholstery (SWC-100)
  or full upholstery (SWC-110) in 10 fabric and 6 leather options.
- Base finish options are: stainless steel, matte black,
  gloss white, or chrome.
- Chair is available with or without armrests.
- Suitable for home or business settings.
- Qualified for contract use.

CONSTRUCTION
- 5-wheel plastic coated aluminum base.
- Pneumatic chair adjust for easy raise/lower action.

DIMENSIONS
- WIDTH 53 CM | 20.87"
- DEPTH 51 CM | 20.08"
- HEIGHT 80 CM | 31.50"
- SEAT HEIGHT 44 CM | 17.32"
- SEAT DEPTH 41 CM | 16.14"

OPTIONS
- Soft or hard-floor caster options.
- Two choices of seat foam densities:
  medium (1.8 lb/ft3) or high (2.8 lb/ft3)
- Armless or 8 position PU armrests

MATERIALS
SHELL BASE GLIDER
- Cast Aluminum with modified nylon PA6/PA66 coating.
- Shell thickness: 10 mm.
SEAT
- HD36 foam

COUNTRY OF ORIGIN
- Italy
"""

fact_sheet_chair
```

## The Prompt

```
In [9]: prompt = f"""
Your task is to help a marketing team create a
description for a retail website of a product based
on a technical fact sheet.
```

Write a product description based on the information provided in the technical specifications delimited by triple backticks.

The description is intended for furniture retailers, so should be technical in nature and focus on the materials the product is constructed from.

At the end of the description, include every 7-character Product ID in the technical specification.

After the description, include a table that gives the product's dimensions. The table should have two columns. In the first column include the name of the dimension. In the second column include the measurements in inches only.

Give the table the title 'Product Dimensions'.

Format everything as HTML that can be used in a website. Place the description in a <div> element.

```
Technical specifications: ```{fact_sheet_chair}```
```

```
response = get_completion(prompt)
print(response)
```

```
In [11]: display(HTML(response))

Introducing our latest addition to the mid-century inspired office furniture collection - the versatile and stylish office chair. This chair is designed to elevate any workspace, whether it be at home or in a business setting. With a variety of shell colors and base finishes to choose from, you can customize this chair to fit your aesthetic perfectly.
```

The construction of this chair is top-notch, featuring a 5-wheel plastic coated aluminum base for stability and mobility. The pneumatic chair adjust allows for easy raise and lower action, ensuring maximum comfort throughout the day.

Constructed with high-quality materials, the shell base glider is made of cast aluminum with a modified nylon PA6/PA66 coating, providing durability and style. The seat is filled with HD36 foam, offering a comfortable seating experience.

Product ID: SWC-100

Product Dimensions:

WIDTH 20.87"

DEPTH 20.08"

HEIGHT 31.50"

SEAT HEIGHT 17.32"

SEAT DEPTH 16.14"

# Advanced Prompting

# Chain-of-Thought Prompting

Chain-of-Thought (CoT) prompting enables complex reasoning capabilities through intermediate reasoning steps. [1]

For zero-shot CoT, simply adding “Let’s think step by step” also works. [2]

(a) Few-shot

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A:  
*(Output) The answer is 8. X*

(b) Few-shot-CoT

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A:  
*(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4. ✓*

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A: The answer (arabic numerals) is  
*(Output) 8 X*

(d) Zero-shot-CoT (Ours)

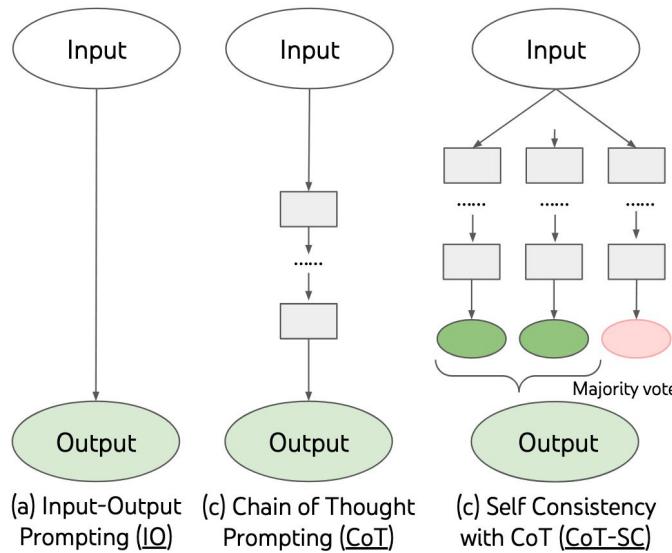
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A: **Let's think step by step.**  
*(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓*

[1] <https://arxiv.org/abs/2201.11903>

[2] <https://arxiv.org/abs/2205.11916>

# Self-Consistency

Sample multiple, diverse reasoning paths through few-shot CoT, and use the generations to select the most consistent answer.



# Demo 1: Prompting

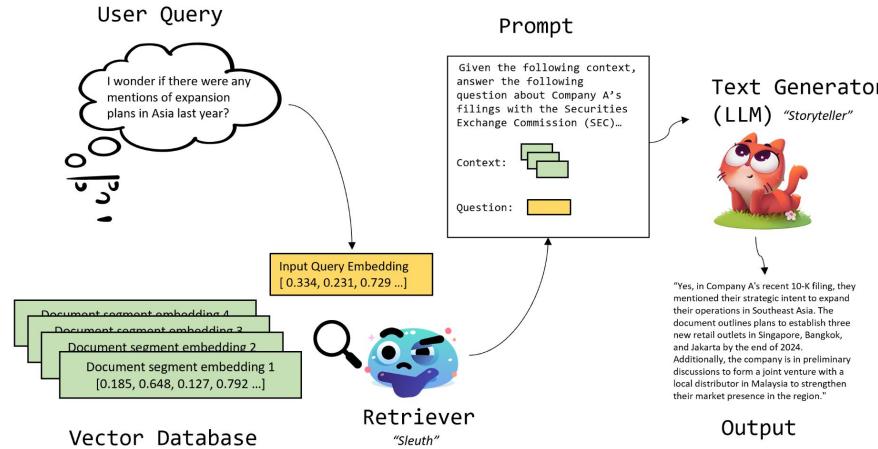
<https://colab.research.google.com/drive/1ppRjCy8b9ZhS5YfK9N3Yxb8iDSuicNzW?usp=sharing>

- Various tasks via many prompting techniques
  - Sentiment analysis
  - QA
  - Few-shot prompting
  - Chain-of-thought prompting
- LLM: Gemini-flash-2.0 (free)

# Retrieval Augmented Generation (RAG)

# Retrieval-augmented Generation (RAG)

- Combining a retriever with an LLM to generate highly informed and contextually relevant outputs.
  - Given a query, the retriever finds the most relevant documents.
  - The LLM takes the documents to answer the query.



# Text Retrieval System

A system that finds the most relevant text (such as an answer, paragraph, or passage) given a query (which could be a question, keywords, or any relevant text).

## Sparse embedding

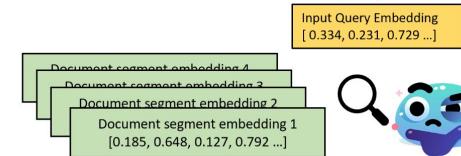
```
{'mil': 0.27, '##vus': 0.39, 'is': 0.16, 'a': 0.11, 'vector': 0.41, 'database': 0.24,  
'built': 0.17, 'for': 0.09, 'scala': 0.19, '##ble': 0.11, 'similarity': 0.30, 'search': 0.19}
```

## BM25

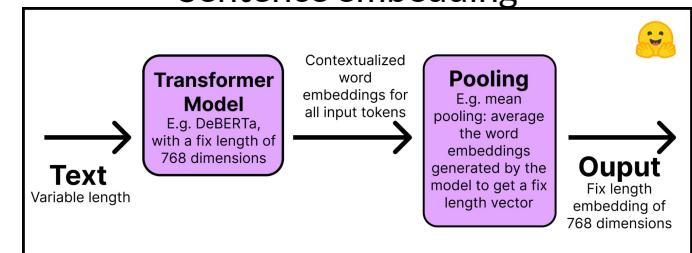
$$BM25 = \sum_{t \in q} \log \left[ \frac{N}{df(t)} \right] \cdot \frac{(k_1 + 1) \cdot tf(t, d)}{k_1 \cdot \left[ (1 - b) + b \cdot \frac{dl(d)}{dl_{avg}} \right] + tf(t, d)}$$

- $k_1, b$  – parameters
- $dl(d)$  – length of document  $d$
- $dl_{avg}$  – average document length

## Dense embedding



## Sentence embedding



# Why use RAG?

- Lets you use your own data. This includes:
  - private data
  - new data
  - domain-specific data
- Reduces hallucinations
  - by grounding the response in factual data
  - improves performance for knowledge-intensive tasks



 What's the capital of Mars?  

 The capital of Mars is Muskland.

# RAG vs Fine-tuning

- Use RAG if data is dynamic.
- Use fine-tuning if you want to change the behaviour of the model.
- You can also use both!

## Common use cases

Some common use cases where fine-tuning can improve results:

- Setting the style, tone, format, or other qualitative aspects
- Improving reliability at producing a desired output
- Correcting failures to follow complex prompts
- Handling many edge cases in specific ways
- Performing a new skill or task that's hard to articulate in a prompt

One high-level way to think about these cases is when it's easier to "show, not tell". In the sections to come, we will explore how to set up data for fine-tuning and various examples where fine-tuning improves the performance over the baseline model.

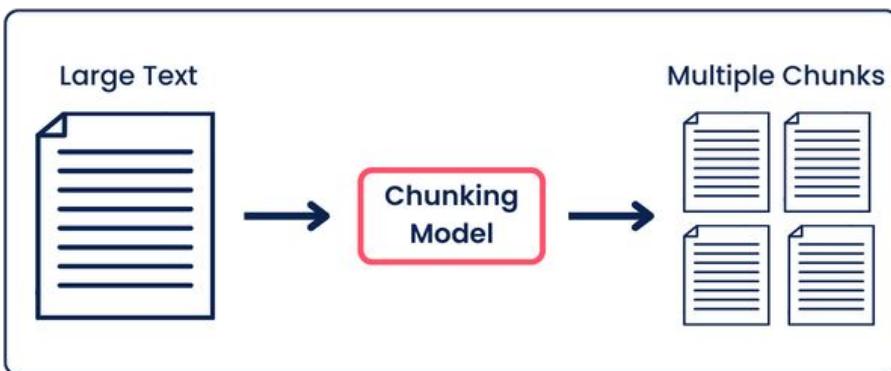
Another scenario where fine-tuning is effective is reducing cost and/or latency by replacing a more expensive model like `gpt-4o` with a fine-tuned `gpt-4o-mini` model. If you can achieve good results with `gpt-4o`, you can often reach similar quality with a fine-tuned `gpt-4o-mini` model by fine-tuning on the `gpt-4o` completions, possibly with a shortened instruction prompt.

<https://platform.openai.com/docs/guides/fine-tuning>

# Typical RAG Workflow

Step 1: Split your documents into manageable-sized chunks.

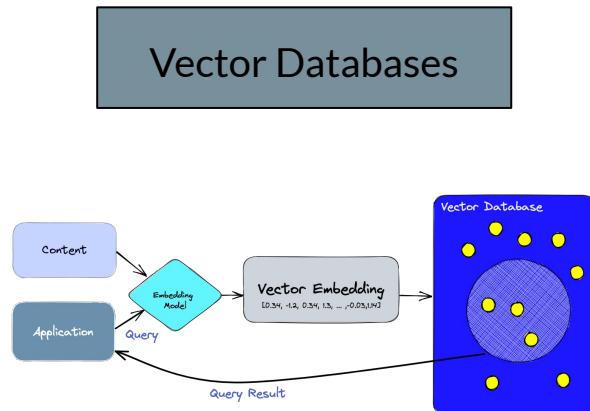
- You will need to find a good chunk size based on your documents.



Name	Classes	Splits On	Adds Metadata	Description
Recursive	RecursiveCharacterTextSplitter, RecursiveJsonSplitter	A list of user defined characters		Recursively splits text. This splitting is trying to keep related pieces of text next to each other. This is the recommended way to start splitting text.
HTML	HTMLHeaderTextSplitter, HTMLSectionSplitter	HTML specific characters	<input checked="" type="checkbox"/>	Splits text based on HTML-specific characters. Notably, this adds in relevant information about where that chunk came from (based on the HTML)
Markdown	MarkdownHeaderTextSplitter	Markdown specific characters	<input checked="" type="checkbox"/>	Splits text based on Markdown-specific characters. Notably, this adds in relevant information about where that chunk came from (based on the Markdown)
Code	many languages	Code (Python, JS) specific characters		Splits text based on characters specific to coding languages. 15 different languages are available to choose from.
Token	many classes	Tokens		Splits text on tokens. There exist a few different ways to measure tokens.
Character	CharacterTextSplitter	A user defined character		Splits text based on a user defined character. One of the simpler methods.
[Experimental] Semantic Chunker	SemanticChunker	Sentences		First splits on sentences. Then combines ones next to each other if they are semantically similar enough. Taken from <a href="#">Greg Kamradt</a>
AI21 Semantic Text Splitter	AI21SemanticTextSplitter		<input checked="" type="checkbox"/>	Identifies distinct topics that form coherent pieces of text and splits along those.

# Typical RAG Workflow

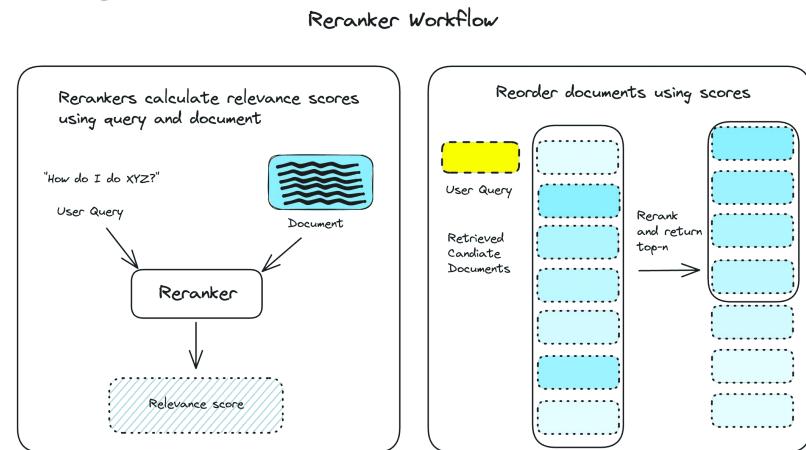
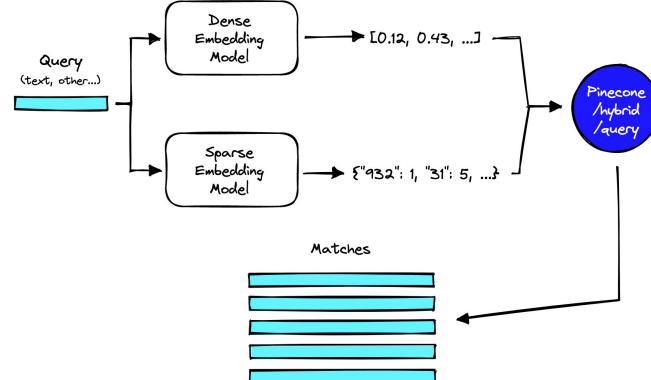
Step 2: Index your chunks into a database or knowledge base of some kind.



# Typical RAG Workflow

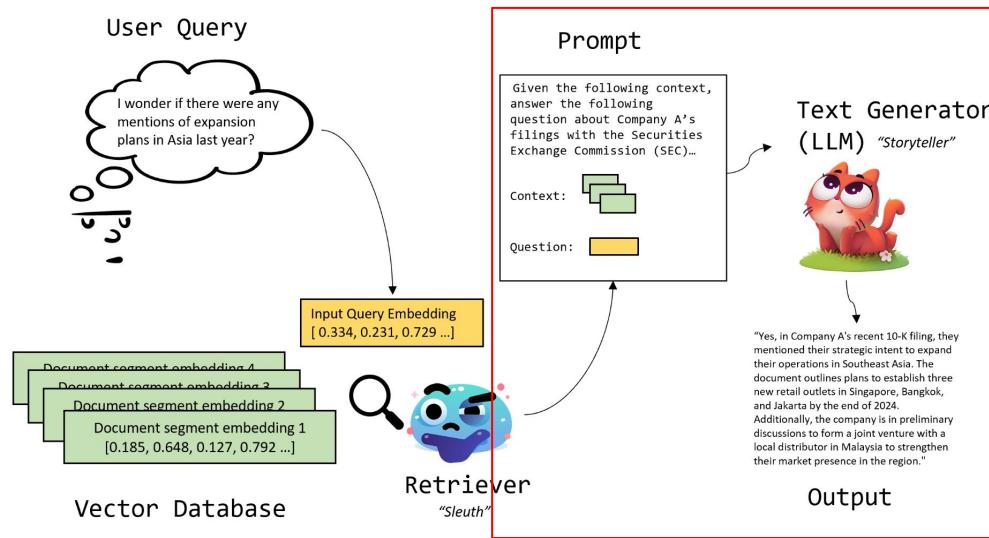
Step 3: Given a query, find the most similar documents in your knowledge base. E.g., perform vector search in a vector database.

- Use hybrid search for better performance
- (Optional) - perform document reranking



# Typical RAG Workflow

Step 4: Feed the query and documents to the LLM to generate an answer.



# Tools & Framework

## Frameworks



**microsoft/graphrag**  
A modular graph-based Retrieval-Augmented Generation (RAG) system

A GitHub repository card for microsoft/graphrag. It shows the repository name, a brief description, and various metrics: 59 stars, 227 forks, 204 discussions, and 21k contributors. A progress bar indicates 100% completion.



## Vector



**Chroma**

**facebookresearch/faiss**



A library for efficient similarity search and clustering of dense vectors.

A GitHub repository card for facebookresearch/faiss. It shows the repository name, a brief description, and various metrics: 167 stars, 4k forks, 225 discussions, and 32k contributors. A progress bar indicates 100% completion.



## Load data and build an index

In the same folder where you created the `data` folder, create a file called `starter.py` file with the following:

```
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader

documents = SimpleDirectoryReader("data").load_data()
index = VectorStoreIndex.from_documents(documents)
```

This builds an index over the documents in the `data` folder (which in this case just consists of the essay text, but could contain many documents).

Your directory structure should look like this:

```
└── starter.py
    └── data
        └── paul_graham_essay.txt
```

## Query your data

Add the following lines to `starter.py`

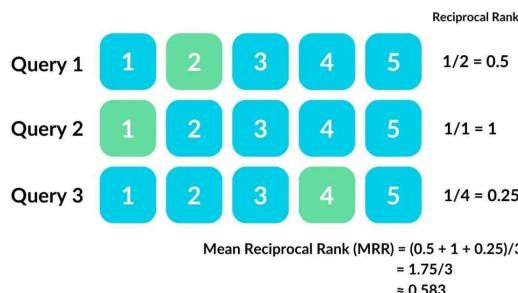
```
query_engine = index.as_query_engine()
response = query_engine.query("What did the author do growing up?")
print(response)
```

This creates an engine for Q&A over your index and asks a simple question. You should get back a response similar to the following: The author wrote short stories and tried to program on an IBM 1401.

# Evaluation

## Retrieval

- **Hit-rate:** assesses whether the relevant documents are being retrieved.
- **MRR (Mean reciprocal rank):** gauges if the relevant documents are ranked high when retrieved.



## Generation

- **Faithfulness:** checks for hallucination (whether the response is grounded to the retrieved contexts.)
- **Relevancy:** evaluates the relevance of both the retrieved context and the generated answer to the initial query.
- **Correctness:** determines if the generated answer is correct based on the reference answer.

# Evaluation



# LangSmith

You can also specify your own metric(s) such as:

- answer conciseness, fluency, etc.; these metrics can be evaluated by a strong LLM like GPT-4o (refer to LLM-as-a-judge).
- or even the number of tokens spent.

The screenshot shows the LangSmith web interface. At the top, there's a header with a parrot icon, a hammer icon, and the text "LangSmith". Below the header, there's a search bar and some navigation links. The main area displays a table of evaluation results for a specific project. The table has columns for "Run Type", "Output", "Latency", "Feedback", "Reference Example", and "Status". There are 17 rows in the table, each representing a different run type (Chain). The "Feedback" column contains scores like "CORRECTNESS: 0.00" and "CORRECTNESS: 1.00". The "Status" column shows green checkmarks for most runs. On the right side of the table, there are filters and search fields for "Full-Text Search", "Feedback", "Feedback Source", "Name", "Run Type", "Status", and "Other".

# LLM as a Judge

[System]

Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user question displayed below. You should choose the assistant that follows the user's instructions and answers the user's question better. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of their responses. Begin your evaluation by comparing the two responses and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[User Question]

{question}

[The Start of Assistant A's Answer]

{answer\_a}

[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]

{answer\_b}

[The End of Assistant B's Answer]

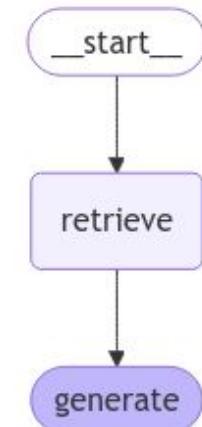
Figure 5: The default prompt for pairwise comparison.

## Demo 2: RAG

<https://colab.research.google.com/drive/1VIUNLaR3vzUhAdSKhh1Sg0tlhxBpcvXW?usp=sharing>

RAG on Reg Chula document (จท. 99)

- LLM - Gemini-flash-2.0
- Retriever - BGE-M3

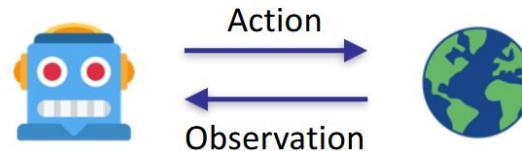


# Agentic LLM

# What is “Agent”

An “intelligent” system that interacts with some “environment”

- Physical environments: autonomous cars, robots
- Digital environments: Siri, AlphaGo



# AI Agent Example

## ELIZA (1966): Symbolic AI agent

---

```
Welcome to
      EEEEEE  LL      IIII    ZZZZZZ  AAAAAA
      EE     LL      II      ZZ  AA   AA
      EEEEEE LL      II      ZZZ  AAAAAAAA
      EE     LL      II      ZZ  AA   AA
      EEEEEE LLLLLL  IIII ZZZZZZ  AA   AA

Eliza is a mock Rogerian psychotherapist.
The original program was described by Joseph Weizenbaum in 1966.
This implementation by Norbert Landsteiner 2005.

ELIZA: Is something troubling you ?
YOU: Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU: They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU: Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU: He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU: It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

- Domain specific!
- Requires manual design
- Cannot work beyond simple domains

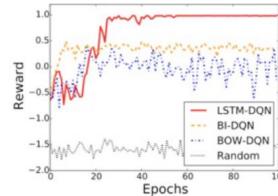
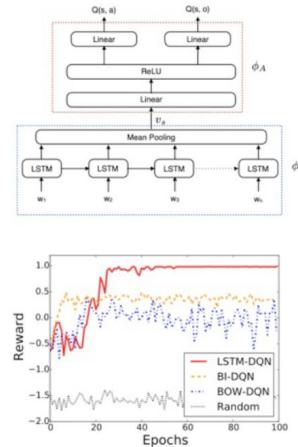
# AI Agent Example

## LSTM-DQN (2015): Deep RL agent

*State 1: The old bridge*  
You are standing very close to the bridge's eastern foundation. If you go east you will be back on solid ground ... The bridge sways in the wind.

*Command: Go east*

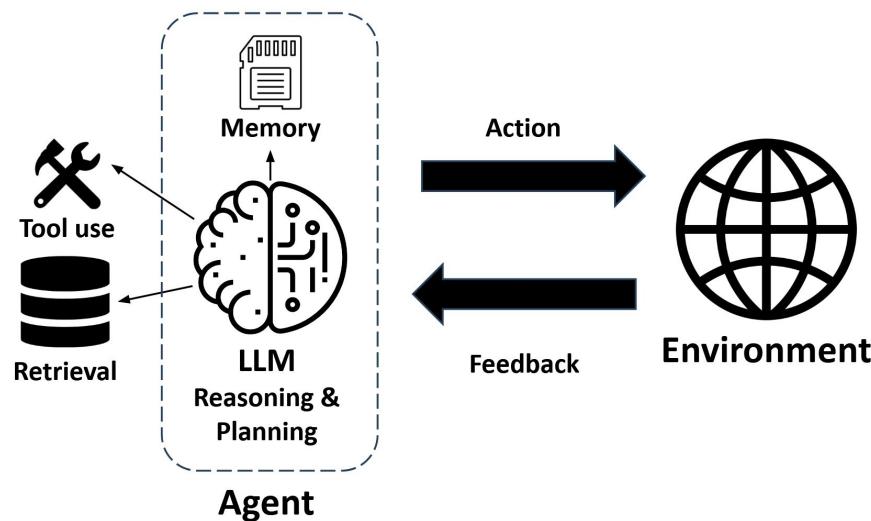
*State 2: Ruined gatehouse*  
The old gatehouse is near collapse. Part of its northern wall has already fallen down ... East of the gatehouse leads out to a small open area surrounded by the remains of the castle. There is also a standing archway offering passage to a path along the old southern inner wall.  
Exits: Standing archway, castle corner, Bridge over the abyss



- Domain specific!
- Requires scalar reward signals
- Requires extensive training

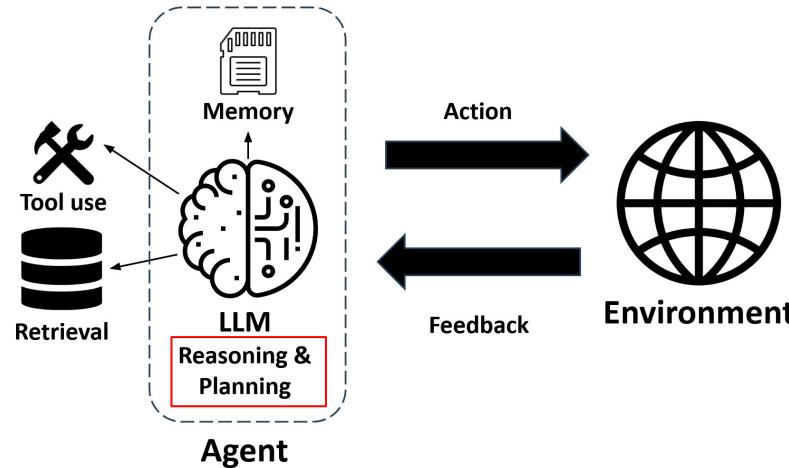
# Agentic LLMs

An agentic LLM is a large language model (LLM) that operates as an autonomous agent, capable of performing tasks, making decisions, and interacting with external tools or APIs to achieve the specified goals.



# Benefit of Agentic LLM

- Enables LLMs to be able to solve complex real-world tasks through:
  - Task decomposition (planning)
  - Allocation of subtasks to specialized modules (tools, expert models, etc)
  - Multi-agent generation (improves responses quality)
- Expands LLMs' capabilities with external tools and knowledge.



# Reasoning - Why do LLMs need reasoning

Humans can learn new tasks with only a handful of examples because **we can reason** [1].

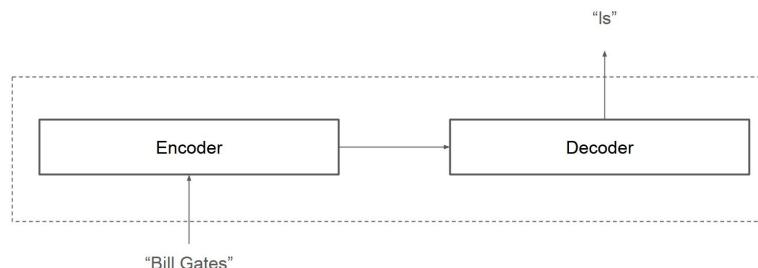
Suppose we have this simple task:

## Last Letter Concatenation

Rule: Take the last letter of each word, and then concatenate them

Input	Output
“Elon Musk”	“nk”
“Bill Gates”	“ls”
“Barack Obama”	?

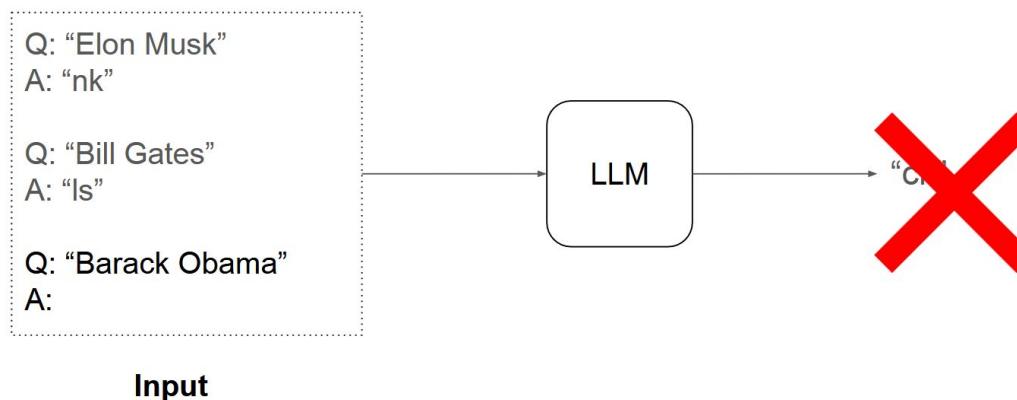
Solve it by ML? Tons of labeled data needed!



# Reasoning - Why do LLMs need reasoning

Simply prompting LLMs with few-shot examples probably won't work as well.

Few-shot prompting for last-letter-concatenation



# Reasoning - Why do LLMs need reasoning

However, giving the reasoning demonstration to the LLM allows it to complete the task easily.

One demonstration is enough, like humans

Q: "Elon Musk"

A: the last letter of "Elon" is "n". the last letter of "Musk" is "k". Concatenating "n", "k" leads to "nk". so the output is "nk".

Q: "Barack Obama"

A: the last letter of "Barack" is "k", the last letter of "Obama" is "a". Concatenating "k", "a" leads to "ka". so the output is "ka".

100% accuracy with only one demonstration example

# Reasoning - How do we achieve reasoning

Key Idea: Derive the Final Answer through **Intermediate Steps**.

**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. X

**Model Input**

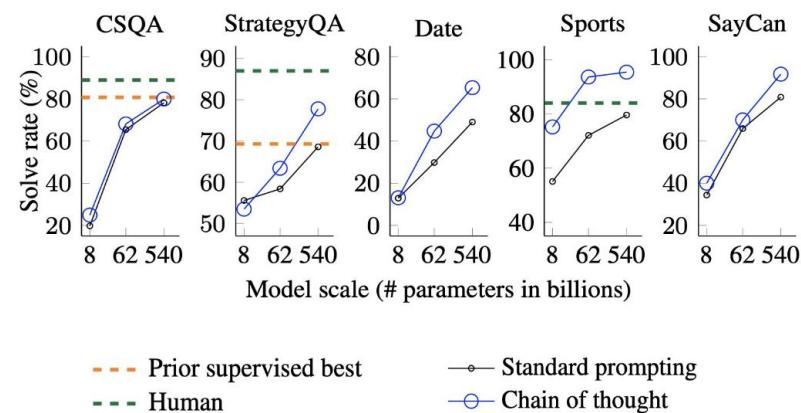
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

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

- Chain-of-Thought (CoT) prompting enables complex reasoning capabilities through intermediate reasoning steps. [1]
- For zero-shot CoT, simply adding “Let’s think step by step” also works. [2]

(a) Few-shot

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A:  
*(Output) The answer is 8. X*

(b) Few-shot-CoT

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A:  
*(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4. ✓*

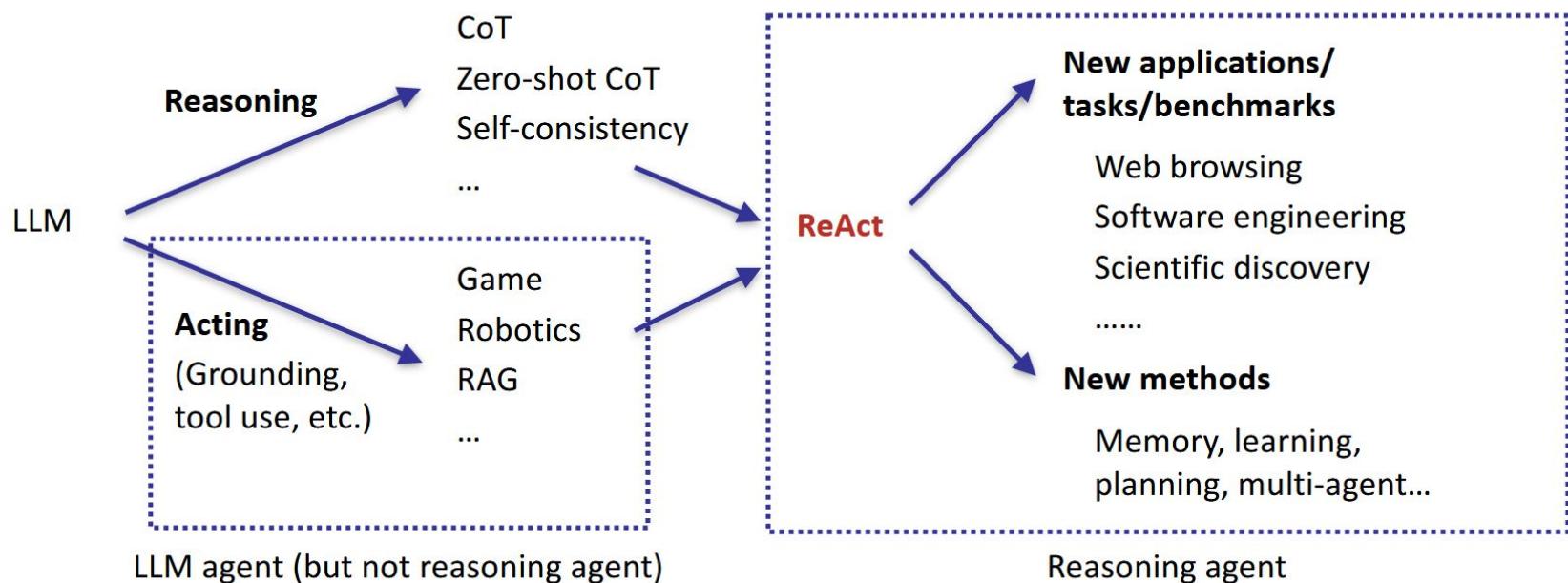
(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A: The answer (arabic numerals) is  
*(Output) 8 X*

(d) Zero-shot-CoT (Ours)

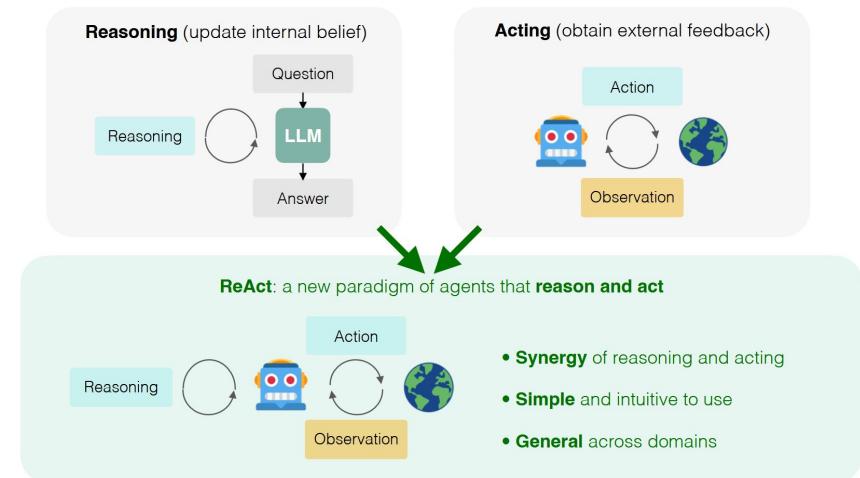
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?  
A: **Let's think step by step.**  
*(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓*

# Reasoning + Acting = ReAct



# ReAct: Synergizing Reasoning and Acting in Language Models

- ReAct = Reasoning + Acting
- Allows LLMs perform **dynamic reasoning** to create, maintain, and adjust plans for acting while also enabling interaction to external environments.
- Unlocks agentic capability.



20

61

# ReAct Example

## Question

Aside from the Apple Remote, what other devices can control the program Apple Remote was originally designed to interact with?

### (1d) ReAct (Reason + Act)

**Thought 1:** I need to search Apple Remote and find the program it was originally designed to interact with.

**Act 1:** `Search[Apple Remote]`

**Obs 1:** The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the `Front Row media center` program ...

**Thought 2:** Apple Remote was originally designed to control the Front Row media center program. I need to search `Front Row` next and find what other device can control it.

**Act 2:** `Search[Front Row]`

**Obs 2:** Could not find `[Front Row]`. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...]

**Thought 3:** Front Row is not found. I need to search `Front Row (software)` .

**Act 3:** `Search[Front Row (software)]`

**Obs 3:** Front Row is a discontinued media center software ...

**Thought 4:** `Front Row (software)` is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.

**Act 4:** `Finish[keyboard function keys]`



# Examples of Agentic Framework

Example 1: Agentic RAG

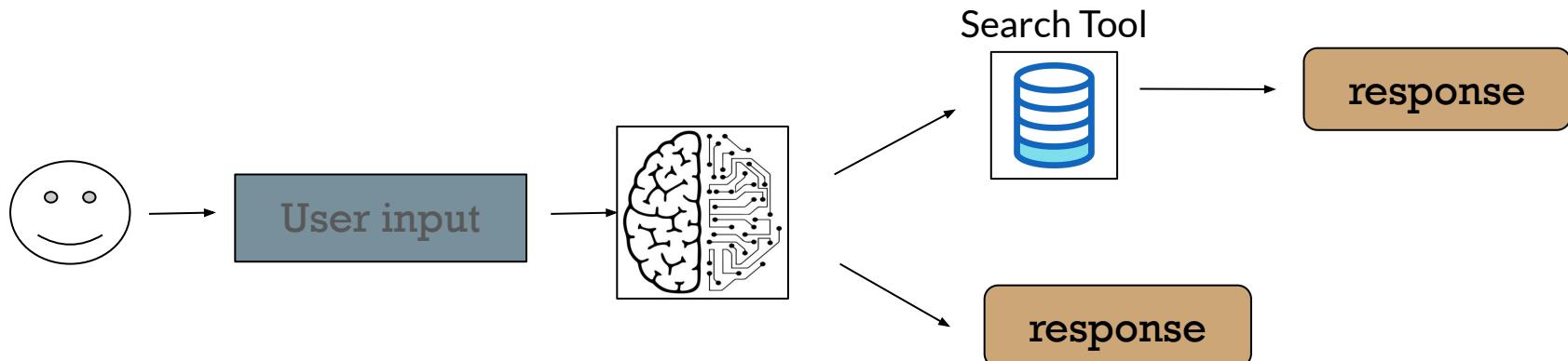
Example 2: Multi-agent

# Example 1: Agentic RAG

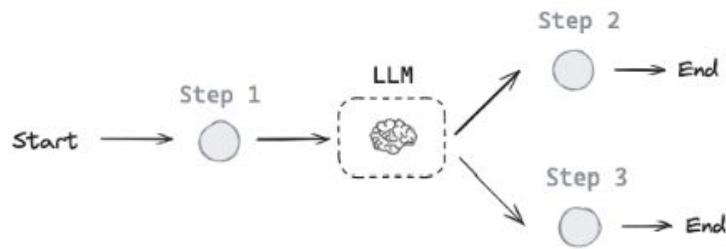
Not all messages need supporting documents for LLMs to respond.

Instead, give the LLM a vector search “tool” (also sometimes referred to as function calling).

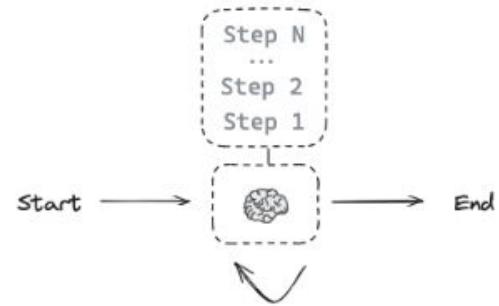
Now it can automatically decide when to use the tool.



# Router

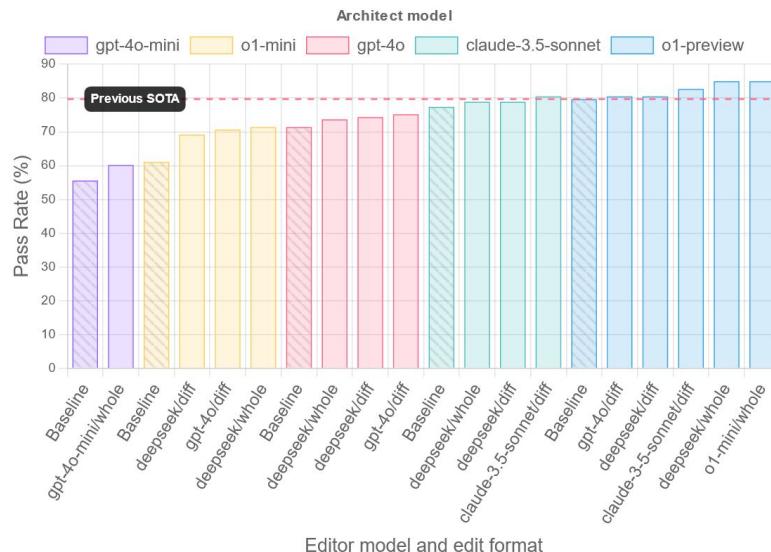


# Fully Autonomous



# Example 2: Multi-agent

Why stop at just a single AI? We can also use multiple of them to complete a task.



```
> /code Add a function to calculate the factorial of a number
Certainly! I'll add a function to calculate the factorial of a number.
Here's the change:
```

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Example usage
print(factorial(5)) # Output: 120
```

This function uses recursion to calculate the factorial. It's been added to your code.

```
> /architect Can we make this simpler?
Yes, you could replace the factorial() function with a call to the math.factorial() builtin. This way you don't have to maintain a custom factorial implementation, and the builtin function is well optimized.
```

Edit the files? (Y)es/(N)o [Yes]: Yes

```
<<<<< SEARCH
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
=====
from math import factorial
>>>> REPLACE
```

# Tools & Framework

Agentic



Monitoring



# Challenges for LLM Agents

- Reasoning and Planning
  - Agents still tend to make mistakes when performing complex tasks.
- Environmental Feedback
  - Can't efficiently recover from mistakes.
  - Can't self-improve
- Safety and Privacy
  - Susceptible to adversarial attacks

# Challenges for LLM Agents

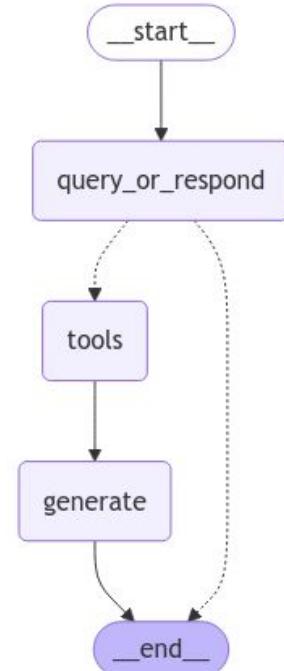
Air Canada has been held liable for a negligent misrepresentation made to a customer by one of its chatbots in a case that one expert said highlights broader risks businesses must consider when adopting AI tools.

# Demo 3 - Agentic RAG

<https://colab.research.google.com/drive/18mGogk7I0ahDCyHD0LX8hJD9vV5TGdMb?usp=sharing>

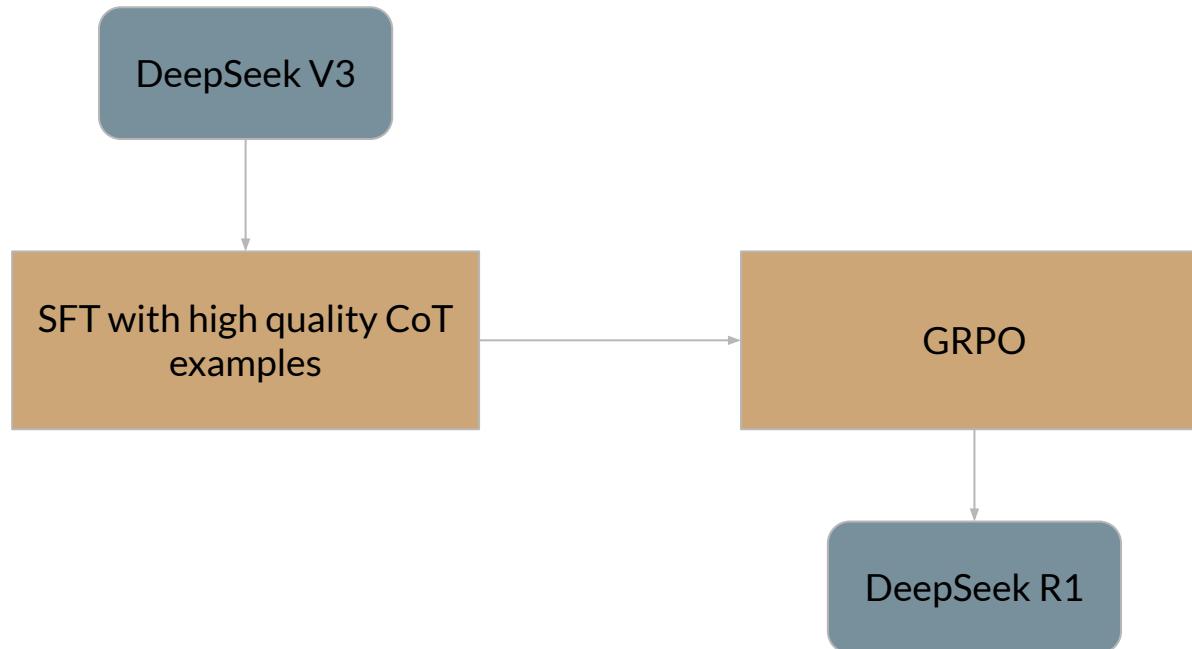
## Agentic RAG with Reg Chula CR99

- LLM - Gemini-flash-2.0
- Retriever - BGE-M3
- LLM as router



Thank you!

# Appendix: DeepSeek R1



# Appendix: DeepSeek R1

## Key points

- Mixture of Experts (MoE) Transformer
- Multihead latent attention
  - Less memory intensive
  - Stronger performance
- Multi-token prediction
  - Improves performance
- GRPO
  - More efficient RL training

Benchmark (Metric)	Claude-3.5-Sonnet-1022	GPT-4o 0513	DeepSeek V3	OpenAI o1-mini	OpenAI o1-1217	DeepSeek R1
Architecture	-	-	MoE	-	-	MoE
# Activated Params	-	-	37B	-	-	37B
# Total Params	-	-	671B	-	-	671B
English	MMLU (Pass@1)	88.3	87.2	88.5	85.2	<b>91.8</b>
	MMLU-Redux (EM)	88.9	88.0	89.1	86.7	-
	MMLU-Pro (EM)	78.0	72.6	75.9	80.3	-
	DROP (3-shot F1)	88.3	83.7	91.6	83.9	90.2
	IF-Eval (Prompt Strict)	<b>86.5</b>	84.3	86.1	84.8	-
	GPQA Diamond (Pass@1)	65.0	49.9	59.1	60.0	<b>75.7</b>
	SimpleQA (Correct)	28.4	38.2	24.9	7.0	<b>47.0</b>
	FRAMES (Acc.)	72.5	80.5	73.3	76.9	-
	AlpacaEval2.0 (LC-winrate)	52.0	51.1	70.0	57.8	-
	ArenaHard (GPT-4-1106)	85.2	80.4	85.5	92.0	-
Code	LiveCodeBench (Pass@1-COT)	38.9	32.9	36.2	53.8	<b>63.4</b>
	Codeforces (Percentile)	20.3	23.6	58.7	93.4	<b>96.6</b>
	Codeforces (Rating)	717	759	1134	1820	<b>2061</b>
	SWE Verified (Resolved)	<b>50.8</b>	38.8	42.0	41.6	48.9
	Aider-Polyglot (Acc.)	45.3	16.0	49.6	32.9	<b>61.7</b>
Math	AIME 2024 (Pass@1)	16.0	9.3	39.2	63.6	<b>79.2</b>
	MATH-500 (Pass@1)	78.3	74.6	90.2	90.0	<b>96.4</b>
	CNMO 2024 (Pass@1)	13.1	10.8	43.2	67.6	-
Chinese	CLUEWSC (EM)	85.4	87.9	90.9	89.9	-
	C-Eval (EM)	76.7	76.0	86.5	68.9	-
	C-SimpleQA (Correct)	55.4	58.7	<b>68.0</b>	40.3	-

Table 4 | Comparison between DeepSeek-R1 and other representative models.