



Large Language Model (LLM)

Professor Peerapon Vateekul, Ph.D.

peerapon.v@chula.ac.th

Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University

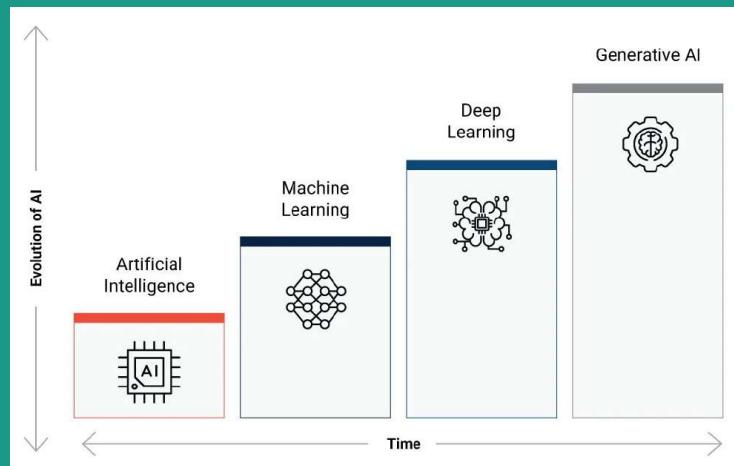


หัวข้อ

- Generative AI: LLM
- Prompting
- RAG
- Agentic LLMs
- LLM Tools
- Other Topics

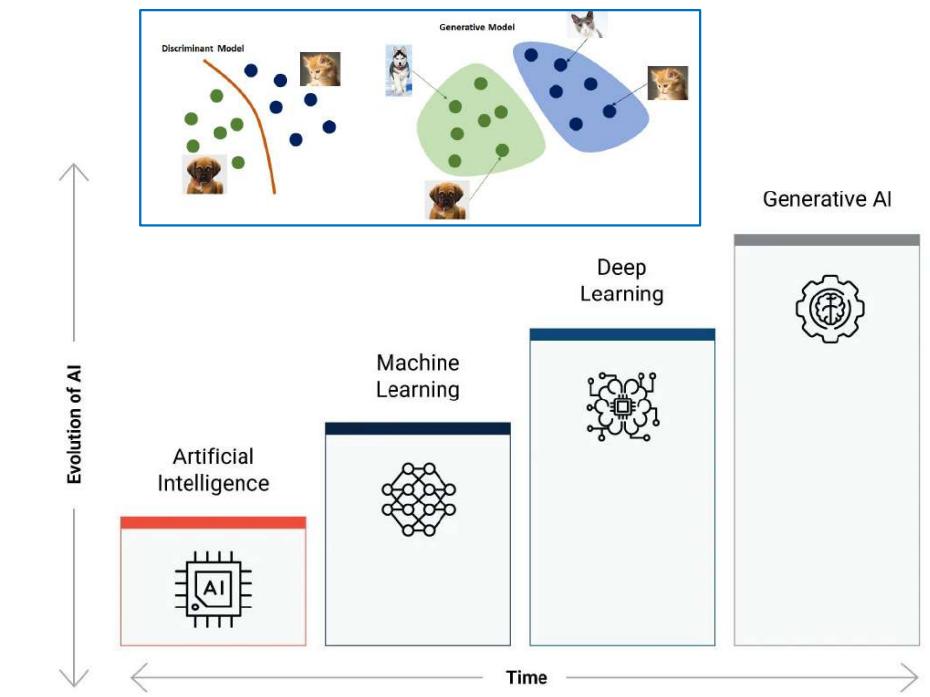
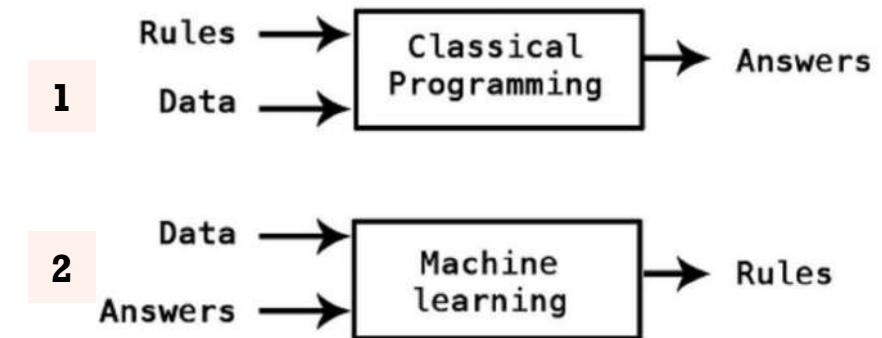
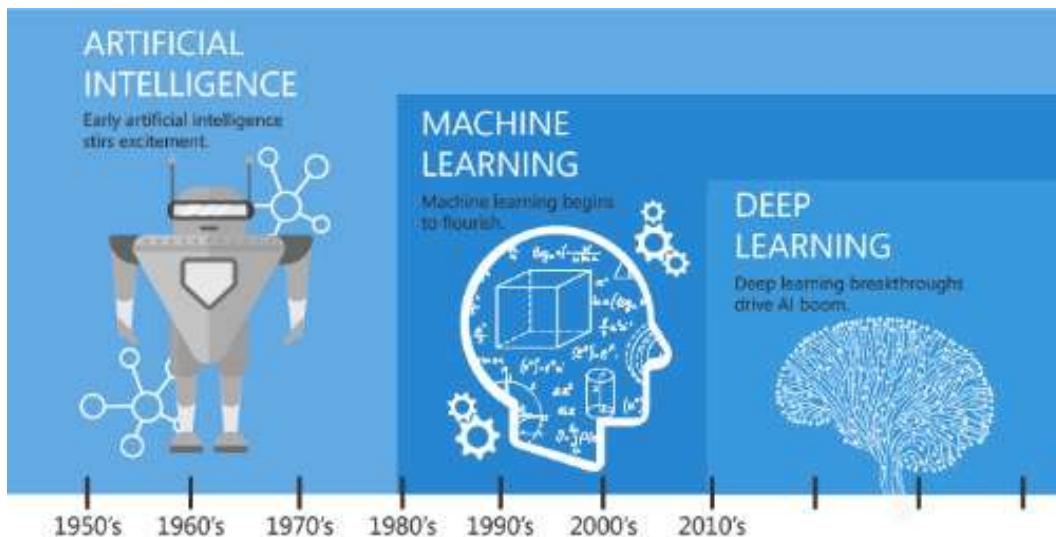
Generative AI: LLM

Credit to Asst. Prof.Ekapol's slide



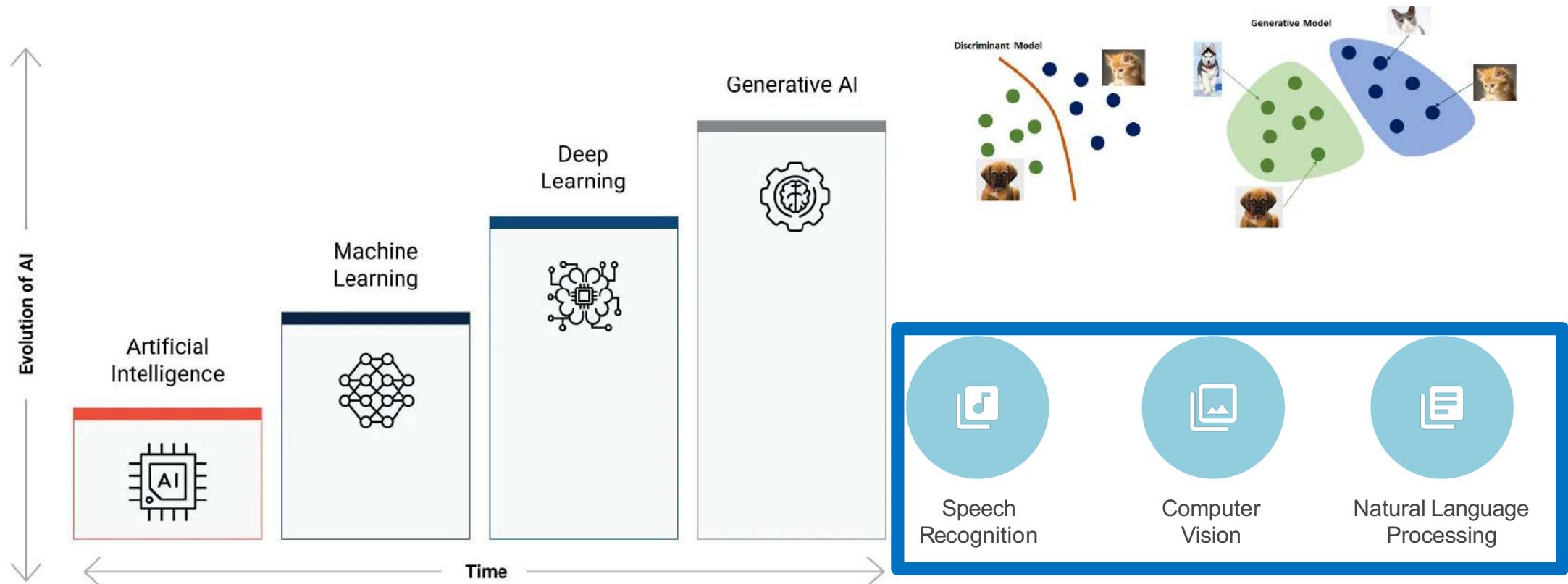
AI = Automation

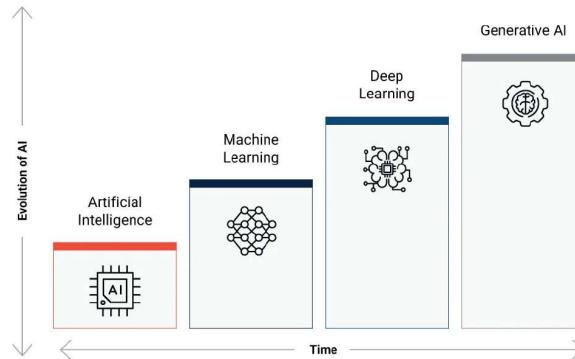
- 1) Rule-based AI
- 2) Machine Learning (ML)



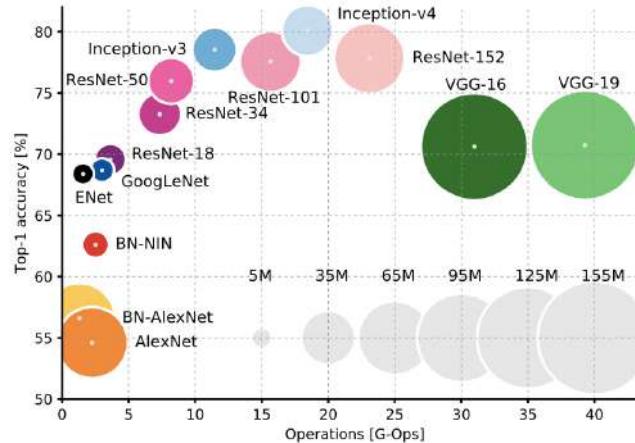
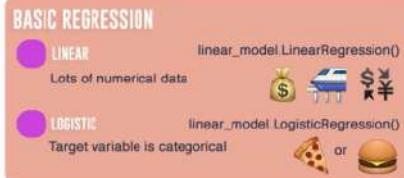
<https://mc.ai/machine-learning-basics-artificial-intelligence-machine-learning-and-deep-learning/>

Recent Research: DL & Generative AI

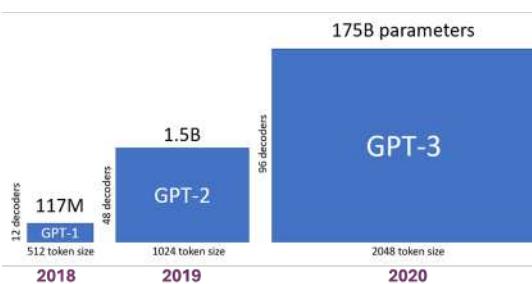




$$\text{Spend} = 500 + 10^* \text{Age} + 20^* \text{Income}_1K$$

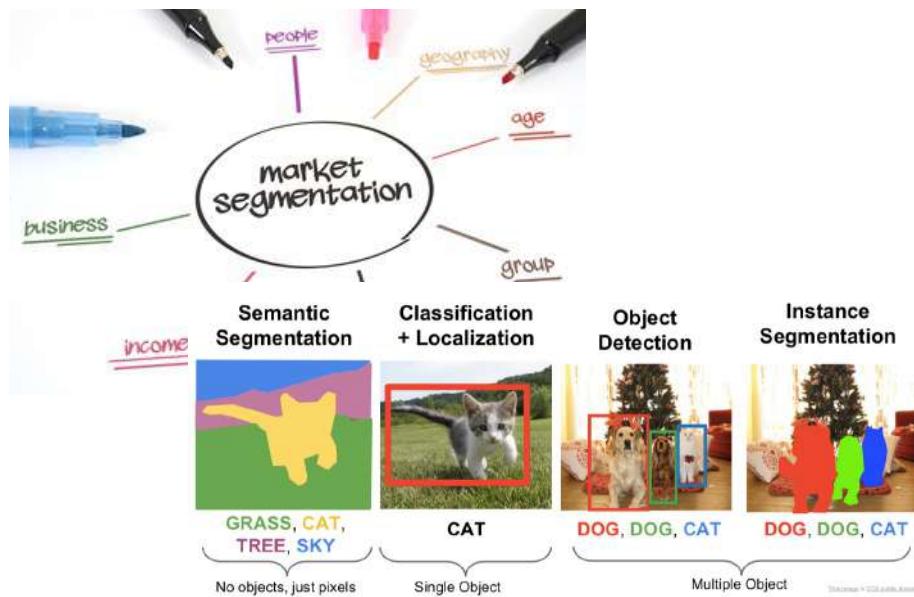


How exactly “Large” it is

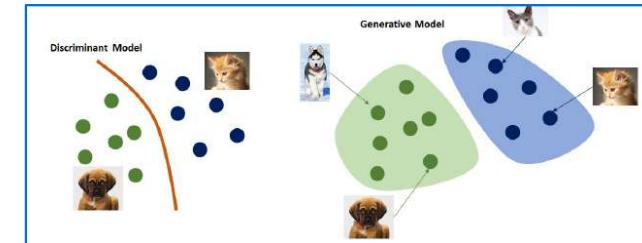


Predictive AI vs. Generative AI

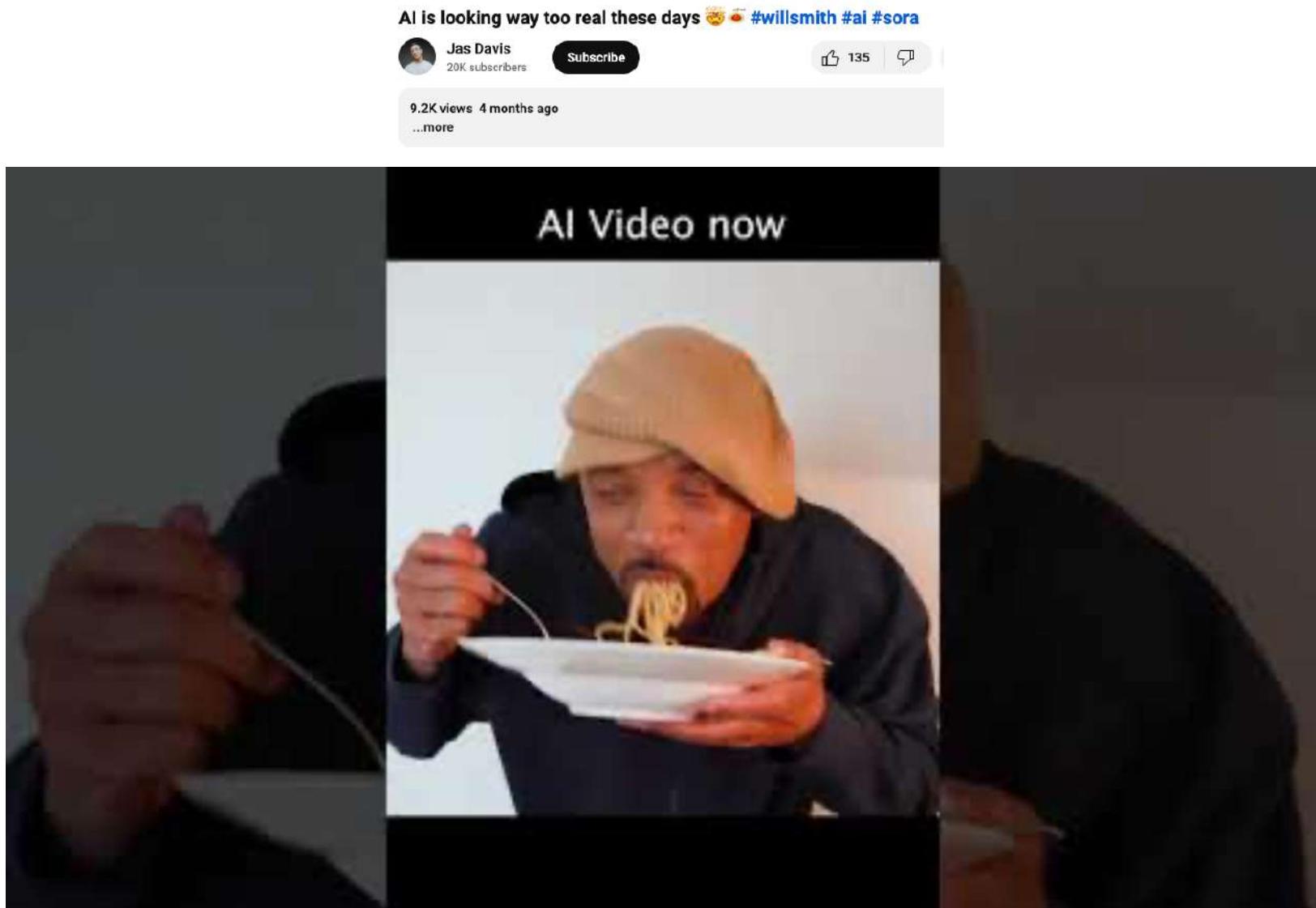
- Predictive AI
 - Predict, Forecast, Detect
 - Cluster, Classify, Segment



- Generative
 - Generate, Generalize



<https://www.youtube.com/watch?v=XQr4Xklqzw8>



<https://openai.com/index/sora/>



ChatGPT

- ChatGPT was launched by OpenAI on 30 Nov 2022.
- ChatGPT is a large language model (LLM) for conversational AI applications.
- Generates human-like text and performs NLP tasks.
- Scalable and flexible for various use cases.
- ChatGPT didn't enclose the details.



OpenAI

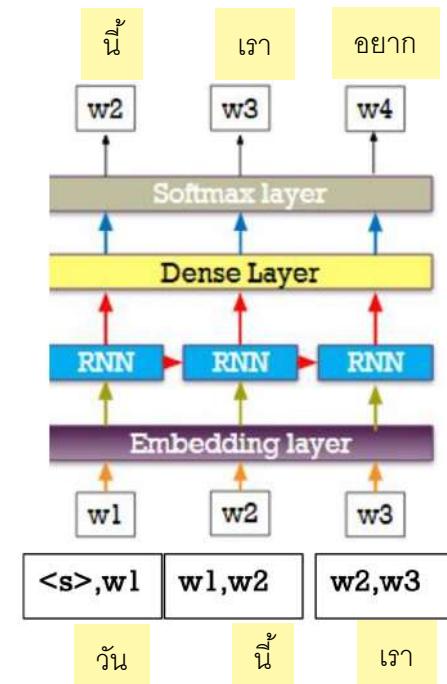


Language Model (LM)

- It is the model that aims to predict next word based on the given previous words.
- So, the model can understand grammar & context.

วันนี่เราอยากจะกินข้าวมัน _____

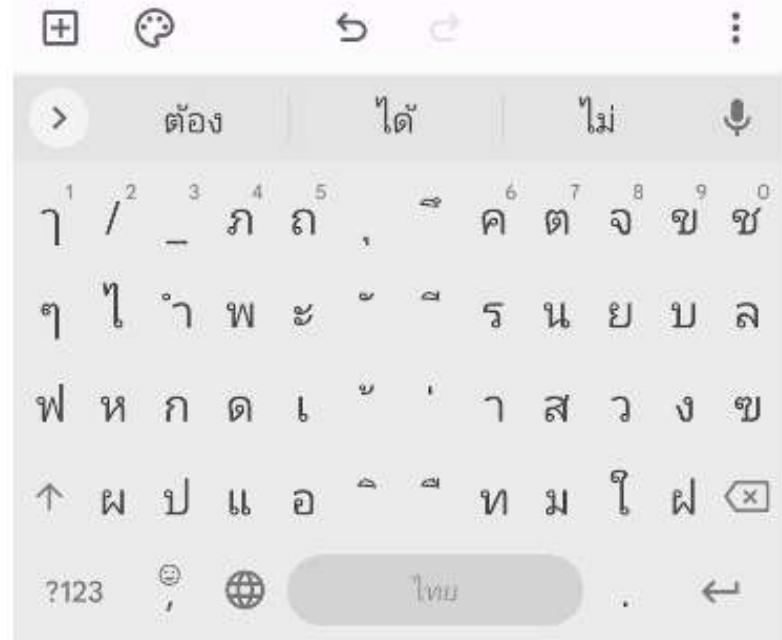
วัน นี่ เรา อยาก จะ กิน ข้าว มัน



สร้างข้อความด้วย AI



สวัสดีวันเป็นผ่านการคัดสรรร่วมกับกลุ่มของท่านจะ



สร้างข้อความด้วย AI

2015

Obama-RNN [2015]

Data: 730,895 tokens (4MB)

Model: 3MB

Good afternoon. God bless you.

The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done. The promise of the men and women who were still going to take out the fact that the American people have fought to make sure that they have to be able to protect our part. It was a chance to stand together to completely look for the commitment to borrow from the American people. And the fact is the men and women in uniform and the millions of our country with the law system that we should be a strong stretches of the forces that we can afford to increase our spirit of the American people and the leadership of our country who are on the Internet of American lives.

Thank you very much. God bless you, and God bless the United States of America.

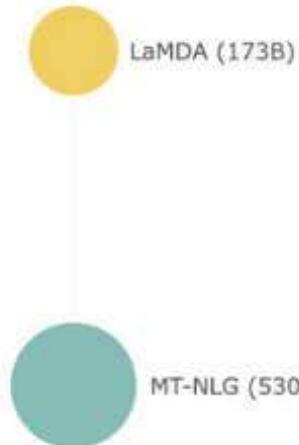
Top Large Language Models

Obama-RNN [2015]
Data: 730,895 tokens (4MB)
Model: 3MB parameters

2020



2021



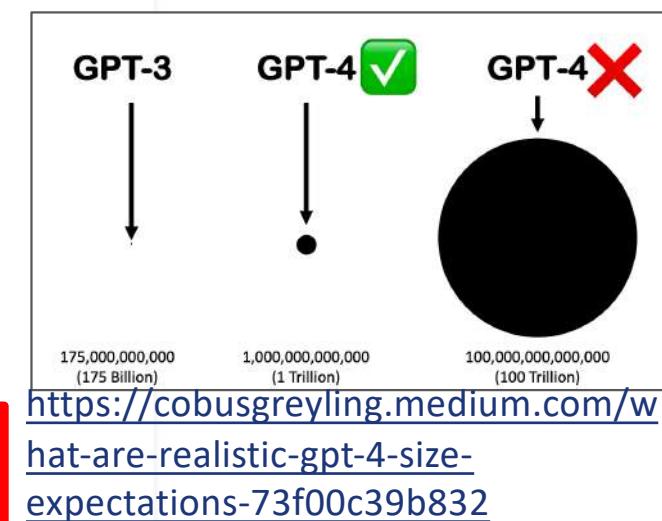
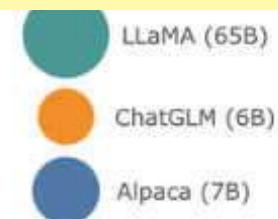
<https://vectara.com/top-large-language-models-langs-gpt-4-llama-gato-bloom-and-when-to-choose-one-over-the-other/>

2022



2023

GPT3 [2020]
Data: ~750GB (30,000x)
Model: 175B parameters (700,000x)
Training cost: \$5M, equivalent to ~300 years

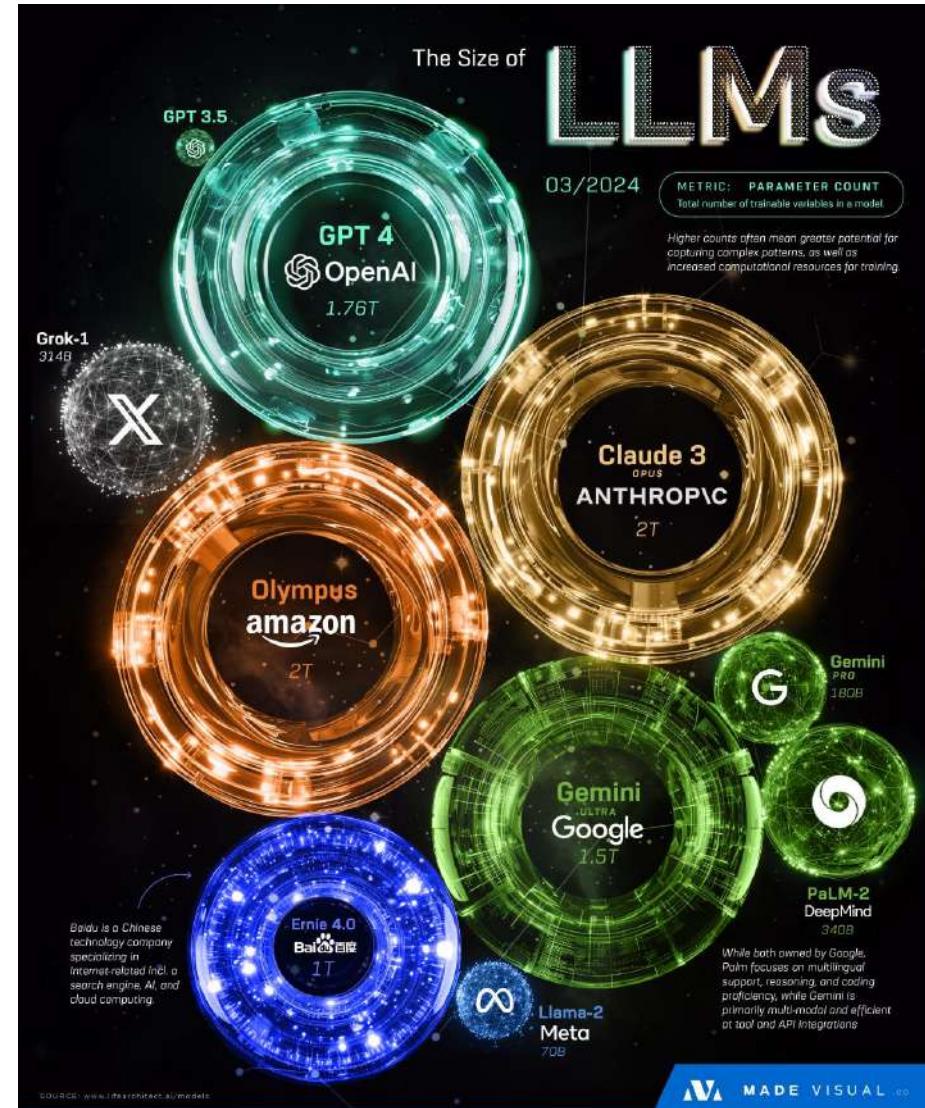


Large + LM = LLM

Model	Year	Parameters (approx)	Tokens Seen (approx)
GPT-1	2018	117M	~0.1B
GPT-2	2019	1.5B	~10B
GPT-3	2020	175B	~300B
GPT-4	2023	~1.7T (MoE est.)	~13T
GPT-5	2025	300B	~114T

Amazon now develops its own family of **foundation models** called **Amazon Nova** (launched late 2024):

- **Nova Micro** → lightweight text-only model (fast, low cost)
- **Nova Lite** → multimodal (text, images, video), efficiency-focused
- **Nova Pro** → stronger multimodal, balance between accuracy & speed
- **Nova Premier** → top-tier multimodal model (teacher model for distillation)
- **Nova Canvas** → image generation
- **Nova Reel** → video generation
- **Nova Sonic** → speech-to-speech (voice)
- **Nova Act** → agent model that can operate tools and browsers

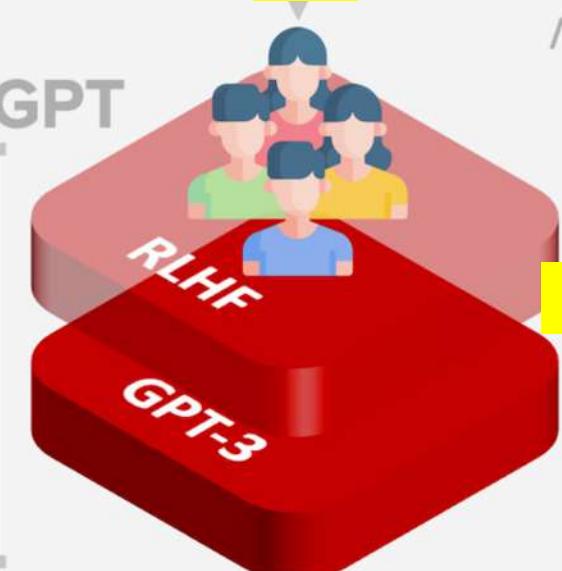


OpenAI



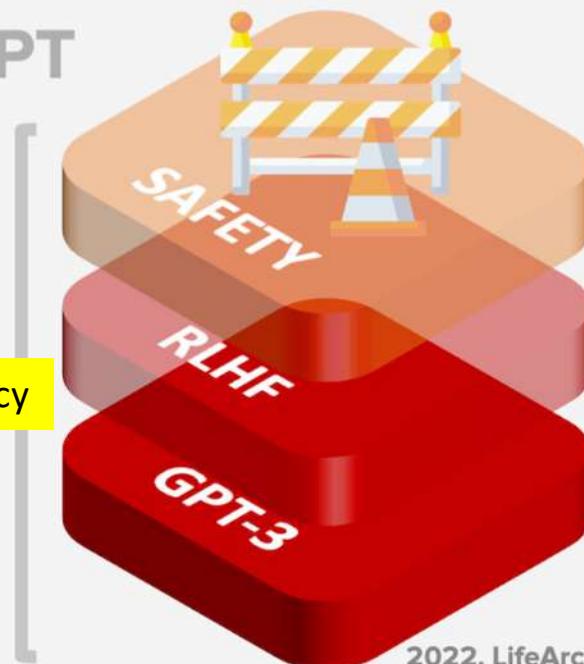
InstructGPT

Jan/2022



ChatGPT

Nov/2022



2022. LifeArchitect.ai

Step 1

Collect demonstration data and train a supervised policy.

LM

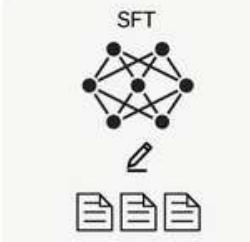
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3.5 with supervised learning.



Step 2

Collect comparison data and train a reward model.

Reward Model (Scoring Model)

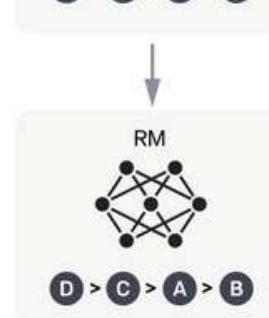
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

RL

A new prompt is sampled from the dataset.



The PPO model is initialized from the supervised policy.



calculates a reward for the output.



The reward is used to update the policy using PPO.

r_k

LLM Capabilities



ChatGPT

1) Assistant Chatbot

A screenshot of the ChatGPT interface. A user message at the top says: "Explain what ChatGPT is to me, but imagine I'm a little high." Below it, a large AI-generated response begins with "L" and includes a small AI icon.

2) Text Summarization

A screenshot of the SCISPACE platform. On the left, there's a sidebar with various icons. In the center, a paper titled "CLIMate: A Multimodal Benchmark for Weather and Climate Events Forecasting" by Haobo Li, Zhaowei Wang, Jiachen Wang, and YueYu Wang from Hong Kong University of Science and Technology is displayed. The abstract discusses the challenges of forecasting weather and climate events. On the right, a summary of the abstract is shown, with the heading "Understanding the Abstract of CLIMate". The summary highlights the importance of forecasting weather and climate events to reduce environmental hazards and minimize potential losses. It also mentions the introduction of WCFP (Weather and Climate Event Forecasting) as a new task for leveraging both numerical meteorological data and textual event data to predict weather and climate events.

[v2] [cs.LG] 16 Feb 2025

3) Content Creation

A screenshot of the Jasper AI Script tool. The main window shows a "Video Script for Jasper AI for Marketing". The script includes sections like "Opening (0:00 - 0:15)", "Introduction to Jasper AI (0:15 - 0:45)", and "Key Features (0:45 - 1:30)". Each section has associated visual descriptions and narration. To the right, a "Style Guide" panel is open, showing various grammar and punctuation rules with examples and violations from the script.

4) Data Analytics

A screenshot of the Conversational Analytics tool. At the top, there's a search bar and a user profile. Below it, a dashboard with a greeting "Hello, Sean" and a call-to-action "Chat with your data". On the left, a sidebar shows "Recent", "Shared with me", and "Trash" sections. On the right, a "Select data to begin:" section lists several data sources: "[GOLD] Looker Coffee - BigQuery", "[GOLD] Looker Coffee - Looker Core", "Call Center - Looker Core [DEMO]", "(1) Orders, Products, and Customers - Looker Coffee...", "[GOLD] Cymbol Direct Call Center [DEMO]", and "GA4 Sessions - Duet AI - Google Analytics 4 - Duet AI". A "Connect to data" button is at the bottom.

<https://fireflies.ai/blog/ai-text-generators>

AI text generation tools

The image displays three AI text generation tools side-by-side:

- ChatGPT**: A screenshot of the ChatGPT interface. It shows a sidebar with recent AI Prompts and March news items. The main area features a large blue "ChatGPT" button and a message "How can I help you today?". Below the message are two input fields: "Write a spreadsheet formula to convert a date to the weekday" and "Plan an itinerary to experience the wildlife in the Australian outback".
- Google Gemini**: A screenshot of the Google Gemini interface. It features a large blue "Google Gemini" button and a message "Hello, Ayush. How can I help you today?". Below the message are four input suggestions: "Help me compare these college majors", "Help me get organized with a list of 10 tips", "Ideas to surprise a friend on their birthday", and "Help design a database schema for a business".
- Anthropic Claude**: A screenshot of the Anthropic Claude interface. It shows a sidebar with the Claude logo. The main area has a message "What can I help you with?" and a "Message Claude..." input field with a red send button.



May 13, 2024

Hello GPT-4o

We're announcing GPT-4o, our new flagship model that can reason across audio, vision, and text in real time.

[Contributions >](#) [Try on ChatGPT ↗](#) [Try in Playground ↗](#) [Rewatch live demos >](#)

GPT-4o ("o" for "omni") is a step towards much more natural human-computer interaction—it accepts as input any combination of text, audio, image, and video and generates any combination of text, audio, and image outputs. It can respond to audio inputs in as little as 232 milliseconds, with an average of 320 milliseconds, which is similar to human response time in a conversation. It matches GPT-4 Turbo performance on text in English and code, with significant improvement on text in non-English languages, while also being much faster and 50% cheaper in the API. GPT-4o is especially better at vision and audio understanding compared to existing models.

<https://openai.com/index/hello-gpt-4o/>

Commercial Large Language Model (LLM)

Global LLMs: OpenAI GPT, Google Gemini, LLaMA etc.

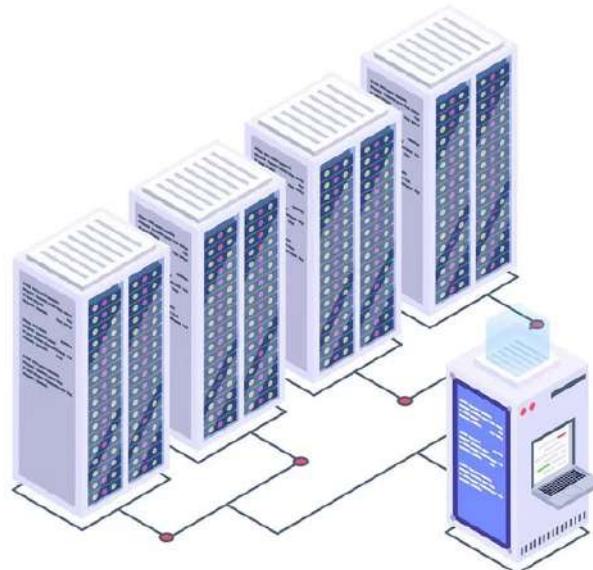


Thai LLMs: OpenThaiGPT, Typhoon, etc.



Loading a Pretrained Model Locally vs. Using a Cloud-based API LLM

LLaMA
by  Meta



LLM Use Cases: Domain Specific Chatbot



Overview Our Goals Partners Explore Try it

MEDICAL FOCUS GENERATIVE AI

PreceptorAI is your pocket 2nd opinion for medical professionals finely tuned to Thai clinical guidelines.

Try it now



The screenshot shows the Thanoy AI Chatbot website. At the top, there's a navigation bar with links to Main Site, AI API, Docs, Support, and Contact Us. Below the navigation, the word "Thanoy" is prominently displayed in large blue letters, with "nuoy" written vertically below it. The text "ผู้ช่วยทนาย AI Chatbot" is also present. On the left, there's a QR code for LINE and a green "Add friend" button. On the right, there's a cartoon character of a fish-headed person in a suit, a red book titled "LAW", and two smartphones displaying the Thanoy app interface.

Prompting

Prompt engineer aims to solve alignment problem.

- Language Model (LM) is originally trained to predict the next word, **NOT** answer the question.
- GPT (GPT3 is 175B parameters) is usually **frozen (not trained)**.
- Since we cannot change the model, we need to align (change) the question (also called prompt).

Input (Prompt)	Output
The patient was died.	The patient's body was found in a dark alley behind the hospital's...
"The patient was died." correct this	claim if you really believe such figures....
Poor English input: The patient was died.	Good English output: The patient died.



Jobs of the Future: AI Prompt Engineer



Cody W Burns

Emerging Technology Visionary | Distributed Systems | Privacy | Executive Leadership

10 articles

+ Follow

October 19, 2022

JOB OF THE FUTURE: AI PROMPT ENGINEER

Cody Burns
25

<https://www.linkedin.com/pulse/jobs-future-ai-prompt-engineer-cody-w-burns/>

Prompt Engineering Techniques



Zero-shot
prompting



Few-shot prompting
or in-context
learning



Chain-of-thought
prompting



Tree-of-thought
prompting



Iterative
refinement



Feedback
loops



Prompt
chaining



Role-playing



Maieutic
prompting



Complexity-based
prompting

servicenow.

<https://www.servicenow.com/ai/what-is-prompt-engineering.html>



How to update LLM knowledge?

- Train from scratch again?
 - Way too expensive
- Continue training on new data?
 - LMs forget old data (catastrophic forgetting)
- In-context learning!

Zero-shot

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

```

1 Translate English to French: ← task description
2 cheese => ..... ← prompt
  
```

One-shot

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

```

1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
  
```

Few-shot

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

```

1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée
4 plush girafe => girafe peluche
5 cheese => ..... ← prompt
  
```

+

In-context Learning (ICL)

Arxiv-2020

EMNLP-2022

EMNLP-2024



1) In-context learning (ICL): Example of ICL (GPT3, Arxiv-2020)

- ICL is an emergent ability of LLMs popularized in the GPT3 paper.
- LLMs can “learn” new tasks from prompts *without training*.

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry
Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan
Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
Benjamin Chess Jack Clark Christopher Berner
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely

05.14165v4 [cs.CL] 22 Jul 2020

<https://arxiv.org/abs/2005.14165>

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.



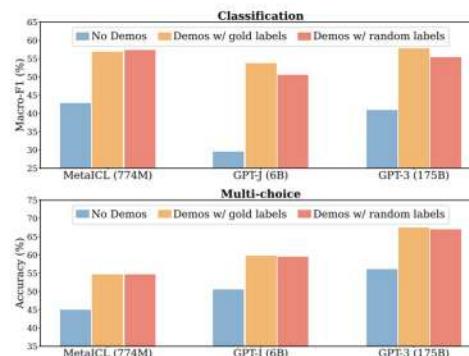
2) In-context learning (ICL): EMNLP-2022

- ICL is hypothesized to be learned during the pretraining stage ([Min et al. \(2022, Best paper at EMNLP\)\[1\]](#)).
- But the LLM is not actually *learning*, i.e., their weights don't change.



Abstract

Large language models (LMs) are able to in-context learn—perform a new task via inference alone by conditioning on a few input-label pairs (demonstrations) and making predictions for new inputs. However, there has been little understanding of *how* the model learns and *which* aspects of the demonstrations contribute to end task performance. In this paper, we show that ground truth demonstrations are in fact not required—randomly replacing labels in the demonstrations barely hurts performance on a range of classification



6 Discussion & Conclusion

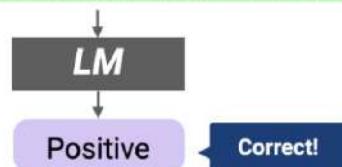
In this paper, we study the role of the demonstrations with respect to the success of in-context learning. We find that the ground truth input-label mapping in the demonstrations matters significantly less than one might think—replacing gold labels with random labels in the demonstrations only marginally lowers the performance. We then identify a series of aspects in the demonstrations and examine which aspect actually contributes to performance gains. Results reveal that (1) gains are mainly coming from *independent* specification of the input space and the label space, (2) the models can still retain up to 95% of performance gains by using either the inputs only or the label set only if the right format is used, and (3) meta-training with an in-context learning objective magnifies these trends. Together, our findings lead to a set of broader indications about in-context learning, as



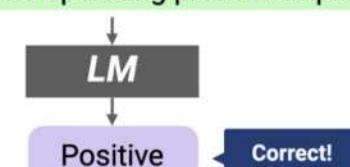
In-context learning (ICL)

- There are many hypotheses on how ICL works:
- [1] showed that ICL acts more as a **pattern recognition procedure**.
 - The **input-output mappings** that are provided allow a model to retrieve similar examples it has been exposed to during training, but the moment you start flipping labels or the template model performance breaks.
- Example 1: Maintain the output labels, but randomly assign the ground truth
- The result is still correct!

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 company anticipated its operating profit to improve. \n _____



Circulation revenue has increased by 5% in Finland. \n Neutral
 Panostaja did not disclose the purchase price. \n Negative
 Paying off the national debt will be extremely painful. \n Positive
 The company anticipated its operating profit to improve. \n _____



[1] <https://arxiv.org/abs/2202.12837>
<https://ai.stanford.edu/blog/understanding-incontext/>

Not the right patterns



In-context learning (ICL)

- There are many hypotheses on how ICL works:
- [1] showed that ICL acts more as a pattern recognition procedure.
 - The input-output mappings that are provided allow a model to retrieve similar examples it has been exposed to during training, but the moment you start flipping labels or the template model performance breaks.
- Example2: Completely random the output labels
- The result is now incorrect!

Circulation revenue has increased by 5% in Finland.

\n Neutral

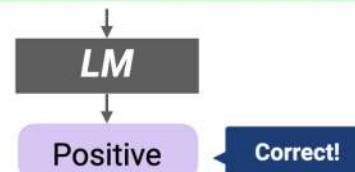
Panostaja did not disclose the purchase price.

\n Negative

Paying off the national debt will be extremely painful.

\n Positive

The company anticipated its operating profit to improve. \n _____



Circulation revenue has increased by 5% in Finland.

\n Unanimity

Panostaja did not disclose the purchase price.

\n Wave

Paying off the national debt will be extremely painful.

\n Guana

The company anticipated its operating profit to improve. \n _____



[1] <https://arxiv.org/abs/2202.12837>

<https://ai.stanford.edu/blog/understanding-incontext/>

+ 3) In-context learning (ICL): EMNLP-2024

33

Still no consensus on how or why ICL works (EMNLP2024)

- Theoretical & empirical (experimental) perspectives.

The Mystery of In-Context Learning: A Comprehensive Survey on Interpretation and Analysis

Yuxiang Zhou[♡], Jiazheng Li[♡], Yanzheng Xiang[♡], Hanqi Yan[♡], Lin Gui[♡], Yulan He^{♡◊}

[♡]Department of Informatics, King's College London [◊]The Alan Turing Institute

{yuxiang.zhou, jiazheng.li, yanzheng.xiang}@kcl.ac.uk

{hanqi.yan, lin.1.gui, yulan.he}@kcl.ac.uk

Abstract

Understanding in-context learning (ICL) capability that enables large language models (LLMs) to excel in proficiency through demonstration examples is of utmost importance. This importance stems not only from the better utilization of this capability across various tasks, but also from the proactive identification and mitigation of potential risks, including concerns regarding truthfulness, bias, and toxicity, that may arise alongside the capability. In this paper, we present a thorough survey on the interpretation and analysis of in-context learning. First, we provide a concise introduction

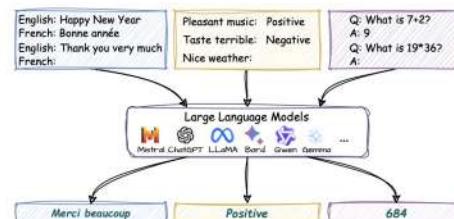


Figure 1: Illustration of In-context Learning.

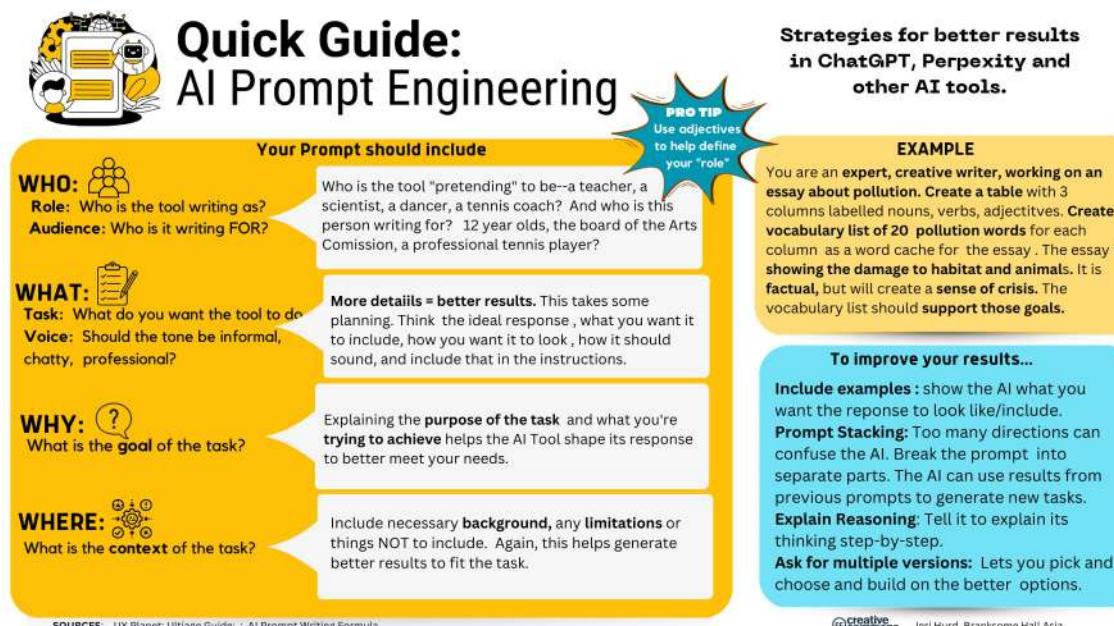
Qwen2 (QwenTeam, 2024) have exhibited proficiency across various tasks, such as machine translation, sentiment analysis, and question answering,

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



In-context learning (ICL): Conclusion

- ICL allows us to use LLMs on any natural language task **without** having to fine-tune the models.
- Instead, we can only rely on supplying LLMs informative prompts.
 - The process of designing prompts is called **prompt engineering**.



+

Prompt Engineering



Prompt types

- 1) **User prompt:** the specific instructions or questions a user provides to an AI system to elicit a desired response.
- 2) **System prompt:** the foundational instructions that dictate an AI's behavior.

```

1 const response = await openai.chat.completions.create({
2   model: "gpt-4o",
3   messages: [
4     {
5       "role": "developer",
6       "content": [
7         {
8           "type": "text",
9           "text": `You are a helpful assistant that answers programming
10             questions in the style of a southern belle from the
11             southeast United States.
12           `
13         }
14       ]
15     },
16   ],
17   "role": "user",
18   "content": [
19     {
20       "type": "text",
21       "text": "Are semicolons optional in JavaScript?"
22     }
23   ],
24 },
25 ],
26 store: true,
27 });

```

```

import anthropic
client = anthropic.Anthropic()

response = client.messages.create(
    model="claude-3-5-sonnet-20241022",
    max_tokens=2048,
    system="You are a seasoned data scientist at a Fortune 500 company.", # <-- role
    messages=[
        {"role": "user", "content": "Analyze this dataset for anomalies: <dataset>{"
    ]
)

print(response.content)

```



Elements of a Prompt

Instruction - a specific task or instruction you want the model to perform

Context - external information or additional context that can steer the model to better responses

Input Data - the input or question that we are interested to find a response for

Output Indicator - the type or format of the output.

+ Example 1

Instruction

Context

Input Data

Output Indicator

Classify the following text into neutral, negative, or positive.

Text: I love this food.

Sentiment:

+ Example 2

Instruction

Context

Input Data

Output Indicator

Answer the question using
information from the given context.

I love this food so much. Everything
is great ...

Question: Did the customer love the
food?

Answer:



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 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)
```

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.

Summarize the text delimited by triple backticks \ into a single sentence.

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: Check for conditions

Task: Convert paragraph 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.\"

```
\\"\\\"{text_1}\\\"\\"
"""

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.\"

```
\\"\\\"{text_2}\\\"\\"
"""

response = get_completion(prompt)
print("Completion for Text 2:")
print(response)
```

Completion for Text 2:

No steps provided.



## Tactic 1.4: Few-shot prompting

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 = f"""
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 = f"""
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 fetch 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, partent en quête d'eau d'un puits au sommet d'une colline, mais rencontrent des malheurs en chemin.",

    "num\_names": 2

}

- **Explicit steps vs. CoT**
- **Action1 → Action2 → ...**
- **Reason1 → Reason2 → ...**

## Key Differences

| Feature   | Explicit Steps in Prompt      | Chain-of-Thought (CoT) Prompting      |
|-----------|-------------------------------|---------------------------------------|
| Purpose   | Guide step-by-step execution  | Encourage step-by-step reasoning      |
| Structure | Direct instructions           | Model generates intermediate thoughts |
| Use Case  | Coding, workflows, automation | Math, logic, problem-solving          |
| Example   | "Step 1: Do X, Step 2: Do Y"  | "Let's think step by step..."         |

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

Without “thinking”

Answer incorrectly as “True”

```
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 + 10x  
Total cost:  $100x + 250x + 100,000 + 10x = 450x + 100,000$

```
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”

Answer correctly as “False”

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 tot
- 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 + 10x  
Total cost:  $100x + 250x + 100,000 + 10x = 450x + 100,000$

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

+

## Advanced Prompting Techniques

+

# 1) Chain-of-Thought Prompting (NeurIPS-2022)

## Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei

Xuezhi Wang

Dale Schuurmans

Maarten Bosma

Brian Ichter

Fei Xia

Ed H. Chi

Quoc V. Le

Denny Zhou

Google Research, Brain Team

{jasonwei, dennyzhou}@google.com

### Abstract

We explore how generating a *chain of thought*—a series of intermediate reasoning steps—significantly improves the ability of large language models to perform complex reasoning. In particular, we show how such reasoning abilities emerge

10 Jan 2023

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

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



# 1) Chain-of-Thought Prompting (NeurIPS-2022)

- 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. ✗

**(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 ✗

**(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>

## 2) Self-Consistency (ICLR-2023)

Published as a conference paper at ICLR 2023

# SELF-CONSISTENCY IMPROVES CHAIN OF THOUGHT REASONING IN LANGUAGE MODELS

Xuezhi Wang<sup>†‡</sup> Jason Wei<sup>†</sup> Dale Schuurmans<sup>†</sup> Quoc Le<sup>†</sup> Ed H. Chi<sup>†</sup>  
Sharan Narang<sup>†</sup> Aakanksha Chowdhery<sup>†</sup> Denny Zhou<sup>†§</sup>

<sup>†</sup>Google Research, Brain Team

<sup>‡</sup>xuezhiw@google.com, <sup>§</sup>dennyyzhou@google.com

### ABSTRACT

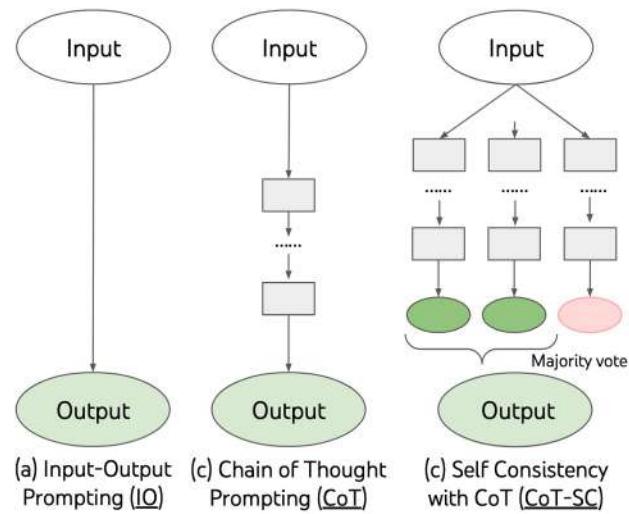
Chain-of-thought prompting combined with pre-trained large language models has achieved encouraging results on complex reasoning tasks. In this paper, we propose a new decoding strategy, *self-consistency*, to replace the naive greedy decoding used in chain-of-thought prompting. It first samples a diverse set of reasoning paths instead of only taking the greedy one, and then selects the most consistent answer by marginalizing out the sampled reasoning paths. Self-consistency leverages the

7 Mar 2023



## 2) Self-Consistency (ICLR-2023)

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



<https://arxiv.org/abs/2203.11171>

Prompt:

```

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: We start with 15 trees. Later we have 21 trees. The difference must be the number of trees they planted. So, they must have planted 21 - 15 = 6 trees. The answer is 6.
Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars. The answer is 5.
Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
A: Leah had 32 chocolates and Leah's sister had 42. That means there were originally 32 + 42 = 74 chocolates. 35 have been eaten. So in total they still have 74 - 35 = 39 chocolates. The answer is 39.
Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?
A: Jason had 20 lollipops. Since he only has 12 now, he must have given the rest to Denny. The number of lollipops he has given to Denny must have been 20 - 12 = 8 lollipops. The answer is 8.
Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?
A: He has 5 toys. He got 2 from mom, so after that he has 5 + 2 = 7 toys. Then he got 2 more from dad, so in total he has 7 + 2 = 9 toys. The answer is 9.
Q: There were nine computers in the server room. Five more computers were installed each day, from monday to thursday. How many computers are now in the server room?
A: There are 4 days from monday to thursday. 5 computers were added each day. That means in total 4 * 5 = 20 computers were added. There were 9 computers in the beginning, so now there are 9 + 20 = 29 computers. The answer is 29.
Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday, he lost 2 more. How many golf balls did he have at the end of wednesday?
A: Michael initially had 58 balls. He lost 23 on Tuesday, so after that he has 58 - 23 = 35 balls. On Wednesday he lost 2 more so now he has 35 - 2 = 33 balls. The answer is 33.
Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
A: She bought 5 bagels for $3 each. This means she spent $15. She has $8 left.
Q: When I was 6 my sister was half my age. Now I'm 70 how old is my sister?
A:

```

Output 1:

When I was 6 my sister was half my age, so she was 3. Now I am 70, so she is  $70 - 3 = 67$ . The answer is 67.

Output 2:

When the narrator was 6, his sister was half his age, which is 3. Now that the narrator is 70, his sister would be  $70 - 3 = 67$  years old. The answer is 67.

Output 3:

When I was 6 my sister was half my age, so she was 3. Now I am 70, so she is  $70/2 = 35$ . The answer is 35.



## 3) ReAct Prompting (ICLR-2023)

Published as a conference paper at ICLR 2023

### REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao<sup>\*1</sup>, Jeffrey Zhao<sup>2</sup>, Dian Yu<sup>2</sup>, Nan Du<sup>2</sup>, Izhak Shafran<sup>2</sup>, Karthik Narasimhan<sup>1</sup>, Yuan Cao<sup>2</sup>

<sup>1</sup>Department of Computer Science, Princeton University

<sup>2</sup>Google Research, Brain team

<sup>1</sup>{shunuy, karthikn}@princeton.edu

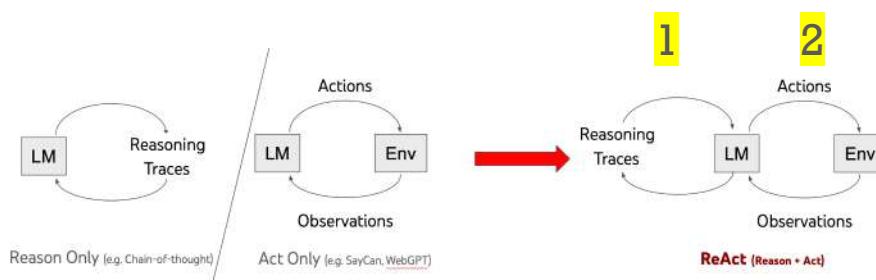
<sup>2</sup>{jeffreyzhao, dianyu, dunan, izhak, yuancao}@google.com

#### ABSTRACT

While large language models (LLMs) have demonstrated impressive performance across tasks in language understanding and interactive decision making, their abilities for reasoning (e.g. chain-of-thought prompting) and acting (e.g. action plan generation) have primarily been studied as separate topics. In this paper, we explore the use of LLMs to generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two: reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions allow it to interface with and gather additional information from external sources such as knowledge bases or environments. We apply our approach, named ReAct, to a diverse set of language and decision making tasks and demonstrate its effectiveness over state-of-the-art baselines in addition to improved human interpretability and trustworthiness. Concretely, on question answering (*HotpotQA*) and fact verification (*Fever*), ReAct overcomes prevalent

## + 3) ReAct Prompting (ICLR-2023)

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



Aside from the Apple Remote, what other devices can control the program, which the 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]`



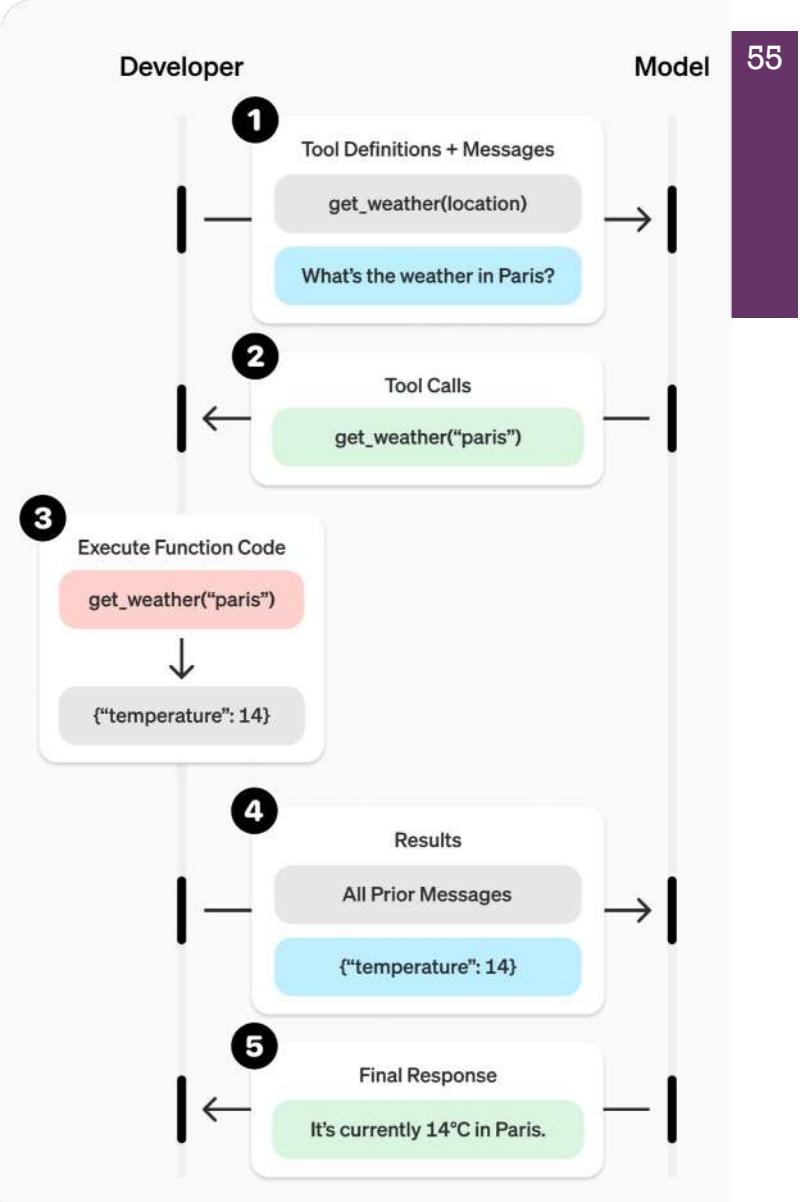


## 4) Tool Use

LLMs nowadays can also use tools (or functions) to improve the quality of their responses when prompted.

Tools can be anything from:

- Web search
- Code interpreter
- And many more



## Tool Use Example 1

56

# ToolFormer (MetaAI, Feb 2023)

## Toolformer: Language Models Can Teach Themselves to Use Tools

Timo Schick Jane Dwivedi-Yu Roberto Dessì<sup>†</sup> Roberta Raileanu  
Maria Lomeli Luke Zettlemoyer Nicola Cancedda Thomas Scialom

Meta AI Research <sup>†</sup>Universitat Pompeu Fabra

### Abstract

Language models (LMs) exhibit remarkable abilities to solve new tasks from just a few examples or textual instructions, especially at scale. They also, paradoxically, struggle with basic functionality, such as arithmetic or factual lookup, where much simpler and smaller models excel. In this paper, we show that LMs can teach themselves to *use external tools* via simple APIs and achieve the best of both worlds. We introduce *Toolformer*, a model trained to decide which APIs to call, when to

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act")]

] 9 Feb 2023

## Tool Use Example 1

57

### ToolFormer (MetaAI, Feb 2023)

Finetuned a GPT-J model (old model, 6B param) to use tools.

Result: Outperforms GPT-3 (old model, 175B)

| Model                 | ASDiv       | SVAMP       | MAWPS       |
|-----------------------|-------------|-------------|-------------|
| GPT-J                 | 7.5         | 5.2         | 9.9         |
| GPT-J + CC            | 9.6         | 5.0         | 9.3         |
| Toolformer (disabled) | 14.8        | 6.3         | 15.0        |
| Toolformer            | <b>40.4</b> | <b>29.4</b> | <b>44.0</b> |
| OPT (66B)             | 6.0         | 4.9         | 7.9         |
| GPT-3 (175B)          | 14.0        | 10.0        | 19.8        |

Table 4: Results for various benchmarks requiring mathematical reasoning. Toolformer makes use of the calculator tool for most examples, clearly outperforming even OPT (66B) and GPT-3 (175B).

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

## Tool Use Example 2

# Program of Thought (PoT) (Google Research, Oct 2023)

Published in Transactions on Machine Learning Research (10/2023)

## Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks

<sup>§</sup>Wenhu Chen

<sup>§</sup>Xueguang Ma

<sup>†</sup>Xinyi Wang

<sup>◦</sup>William W. Cohen

<sup>§</sup>University of Waterloo, Canada

<sup>†</sup>University of California, Santa Barbara, USA

<sup>◦</sup>Google Research, USA

wenhuchen@uwaterloo.ca

x93ma@uwaterloo.ca

xinyi\_wang@ucsb.edu

wcohen@google.com

Reviewed on OpenReview: <https://openreview.net/forum?id=YfZ4ZPt8zd>

### Abstract

Recently, there has been significant progress in teaching language models to perform step-by-step reasoning to solve complex numerical reasoning tasks. Chain-of-thoughts prompting (CoT) is the state-of-art method for many of these tasks. CoT uses language models to produce text describing reasoning, and computation, and finally the answer to a question. Here we propose ‘Program of Thoughts’ (PoT) which uses language models (mainly Codex)

## + Tool Use Example 2

# Program of Thought (PoT) (Google Research, Oct 2023)

Instead of using CoT, PoT outputs reasoning steps in code instead. It then uses the execution result to answer questions

Question: Ketty saves 20000 dollars to the bank. After three years, the sum with compound interest rate is 1000 dollars more than the sum with simple interest rate. What is the interest rate of the bank?

Assuming the interest rate is  $x$ . The sum after two years with simple interest rate is  $20000 + x * 20000 * 3 = 20000 + 60000x$ . The sum after two years with compound interest rate is  $20000 * (1 + x)^3 = 200000 + 60000 * x + 60000x^2 + 20000x^3$ . The difference can be written as  $60000x^2 + 20000x^3 = 1000$ . In order to solve  $x$ , we can use the quadratic formula.  $x = (-b \pm \sqrt{b^2 - 4ac}) / 2a$ , ...,  $x = (-20000 \pm 6160) / 120000$ ,  $x = -0.051333$ .

CoT


  
-0.051333


```
interest_rate = Symbol('x')
sum_in_two_years_with_simple_interest= 20000 +
interest_rate * 20000 * 3
sum_in_two_years_with_compound_interest = 20000 * (1 +
interest_rate)**3
Since compound interest is 1000 more than simple interest.
ans = solve(sum_after_in_yeras_with_compound_interest -
sum_after_two_years_in_compound_interest - 1000,
interest_rate)
```

PoT



python



SymPy


  
 $x = 0.24814$ 


## + Tool Use Example 3

### Tool Use - in ChatGPT and Claude APIs

Available tools are provided to the model along with their names, descriptions, and schemas in the system prompt.

```
tools = [
 {
 "name": "calculator",
 "description": "A simple calculator that performs basic arithmetic operations.",
 "input_schema": {
 "type": "object",
 "properties": {
 "expression": {
 "type": "string",
 "description": "The mathematical expression to evaluate (e.g., '2 + 3 * 4')."
 }
 },
 "required": ["expression"]
 }
 }
]
```



## Tool Use - in ChatGPT and Claude APIs (cont.)

Given a message, the model will choose the tool and output the parameters required for you to run the tool and return the result back to the model. Finally, it will return you its final response.

```
=====
User Message: What is the result of 1,984,135 * 9,343,116?
=====
```

Initial Response:

Stop Reason: tool\_use

Content: [ContentBlock(text='<thinking>\nThe calculator function is the relevant tool to answer this request, since it involves evaluating a mathematical expression.\n\nThe required parameter for the calculator function is:\nexpression: The mathematical expression to evaluate.\n\nThe human has directly provided the full expression to evaluate in their request: "1,984,135 \* 9,343,116". This contains all the information needed for the required expression parameter.\n\nSo I have the necessary information to invoke the calculator tool.\n</thinking>', type='text'), ContentBlockToolUse(id='toolu\_01V2mzqp5qkB5QuRFjJUJLD', input={'expression': '1984135 \* 9343116'}, name='calculator', type='tool\_use')]

Tool Used: calculator

Tool Input: {'expression': '1984135 \* 9343116'}

Tool Result: 18538003464660

[ContentBlock(text='Therefore, the result of 1,984,135 \* 9,343,116 is 18,538,003,464,660.', type='text')]

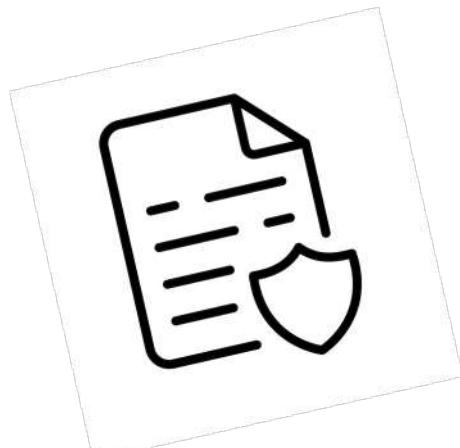
Final Response: Therefore, the result of 1,984,135 \* 9,343,116 is 18,538,003,464,660.

---

RAG

## LLM กับภาพหลอน (hallucination)

---



Private data

 OpenAI  
**GPT-4**



Reliable answer?



Thai language

ChatGPT



ภาพหลอน (hallucination) คือการที่ LLM ตอบแบบให้ข้อมูลที่เหมือนจะจริง (มั่ว) จึงต้องระวังมาก ๆ โดยเฉพาะอย่างยิ่งงานด้านกฎหมาย การแพทย์ เป็นต้น ดังนั้น ก่อนใช้ LLM จึงควรต้องตรวจสอบก่อนใช้เสมอ

พีรพล เวทีกุล คือใคร



พีรพล เวทีกุล เป็นนักข่าวที่มีชื่อเสียงในประเทศไทย โดยเฉพาะในการสื่อสารมวลชน เขายังเป็นที่รู้จักในฐานะผู้ประกาศข่าวทางช่อง 3 และเป็นพิธีกรรายการที่มีความนิยมในหลายรายการโทรทัศน์ เช่น รายการข่าวพิเศษ ต่าง ๆ และรายการสารคดีต่าง ๆ ที่ออกอากาศในช่อง 3 ของประเทศไทย

ဂ ဂ ဂ ဂ ဂ

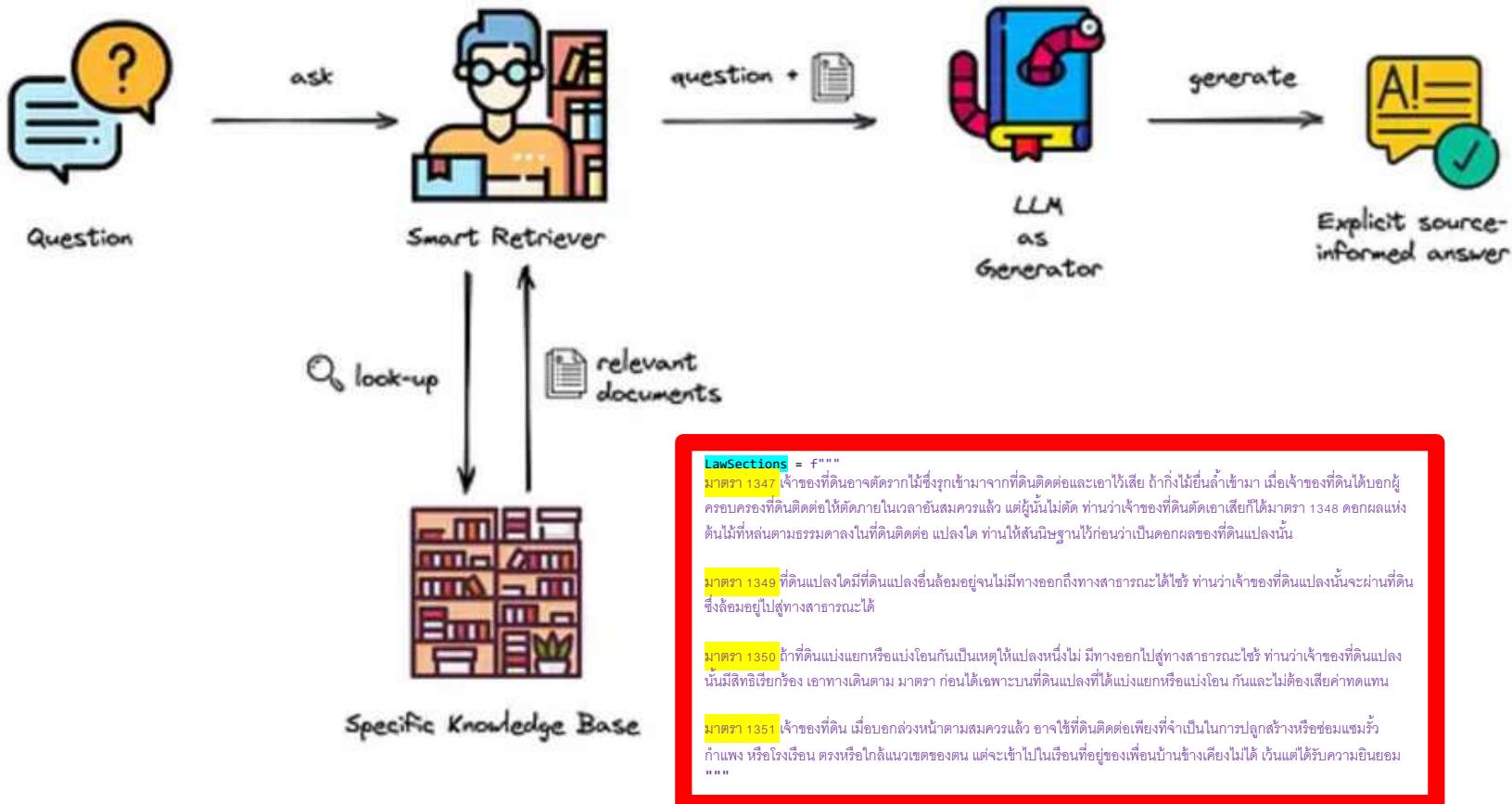
ณรงค์เดช กีรติพราวน์ คือใคร



ณรงค์เดช กีรติพราวน์ เป็นนักการเมืองและนักธุรกิจชาวไทยที่มีชื่อเสียง โดยเป็นสมาชิกพรรคการเมืองชั้นนำ ในประเทศไทย นอกจากนี้เขายังเป็นผู้บริหารในธุรกิจต่าง ๆ

ဂ ဂ ဂ ဂ ဂ

# Retrieval Augmented Generation (RAG)





# Typical RAG Workflow (1)

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

- You will need to find a good chunk size based on your documents.
- If structure matters in your document, try using a markdown splitter.

## Markdown splitter example

For example, if we want to split this markdown:

```
md = '# Foo\n\n## Bar\n\nHi this is Jim \nHi this is Joe\n\n## Baz\n\nHi this is Molly'
```

We can specify the headers to split on:

```
[("#", "Header 1"), ("##", "Header 2")]
```

And content is grouped or split by common headers:

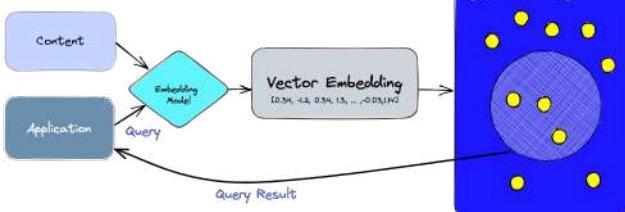
```
{'content': 'Hi this is Jim \nHi this is Joe', 'metadata': {'Header 1': 'Foo', 'Header 2': 'Bar'}}
{'content': 'Hi this is Molly', 'metadata': {'Header 1': 'Foo', 'Header 2': 'Baz'}}
```

| Name                            | Classes                                                  | Splits On                             | Adds Metadata                       | Description                                                                                                                                               |
|---------------------------------|----------------------------------------------------------|---------------------------------------|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Recursive                       | RecursiveCharacterTextSplitter,<br>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,<br>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.                                                                      |

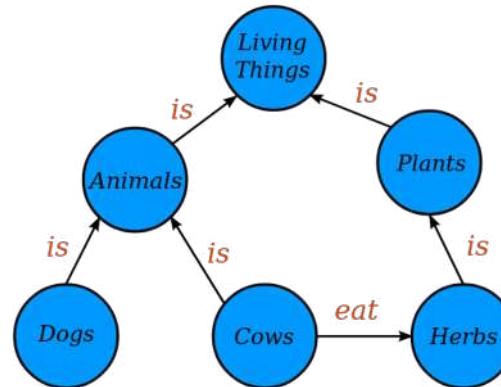
## + Typical RAG Workflow (2)

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

Vector Databases



Knowledge Graphs



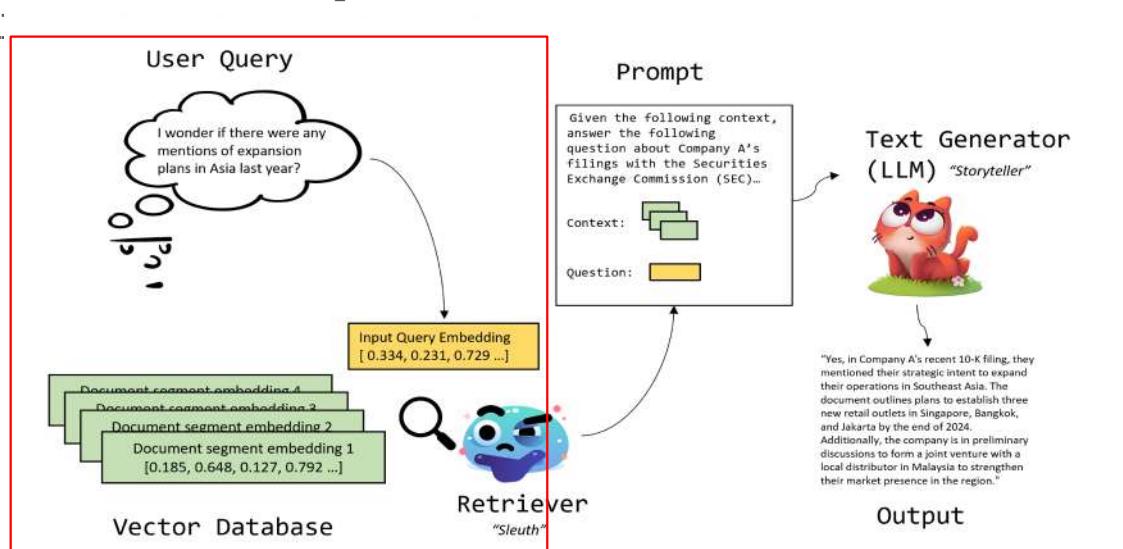
Search Engines



## + Typical RAG Workflow (3)

**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)





## There are 3 search paradigms

(a) is the most efficient but least performance.

(b) is the most computationally expensive but performs best.

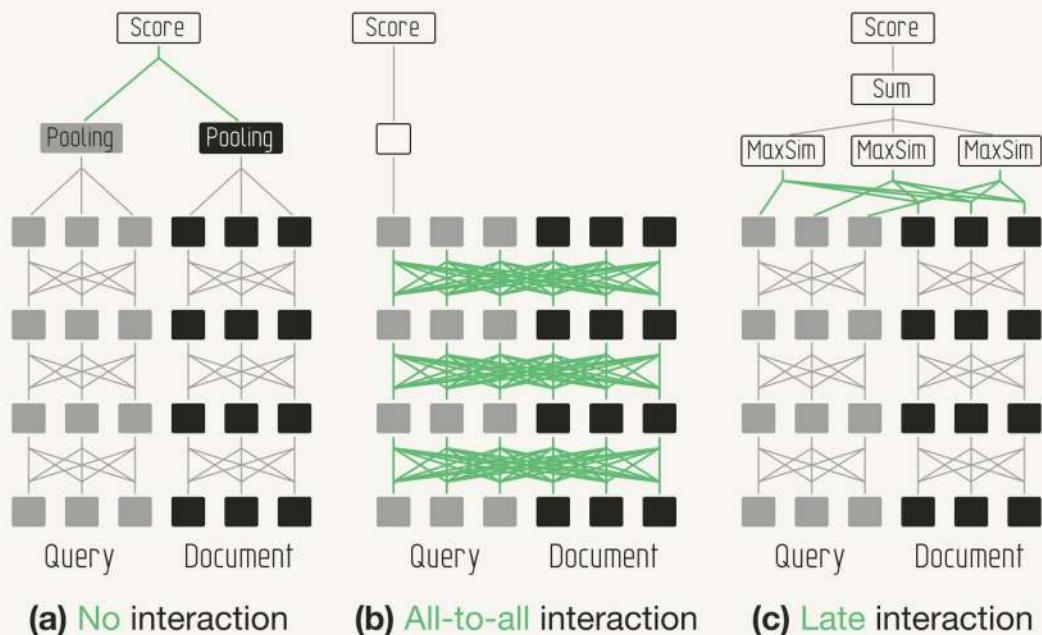
**(a) and (c) can be precomputed offline.**

To trade-off between speed and accuracy, **two-stage ranking** is often used:

- (a) (and optionally (c)) is used to find a small group of candidates
- then run (b) for the final score.

## Types of interaction

Fig. 1: Three types of interaction in dense retrievers.

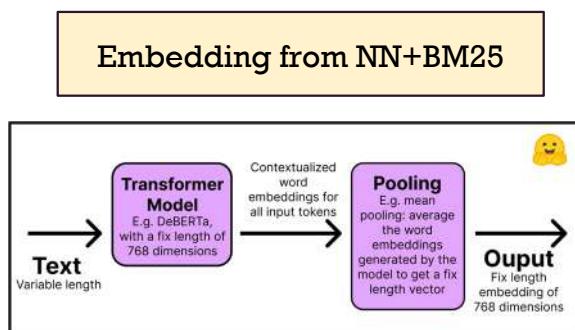




# Hybrid Search

- Step3: IR
- **3.1) Use hybrid search for better performance**
- 3.2) (Optional) - perform document reranking

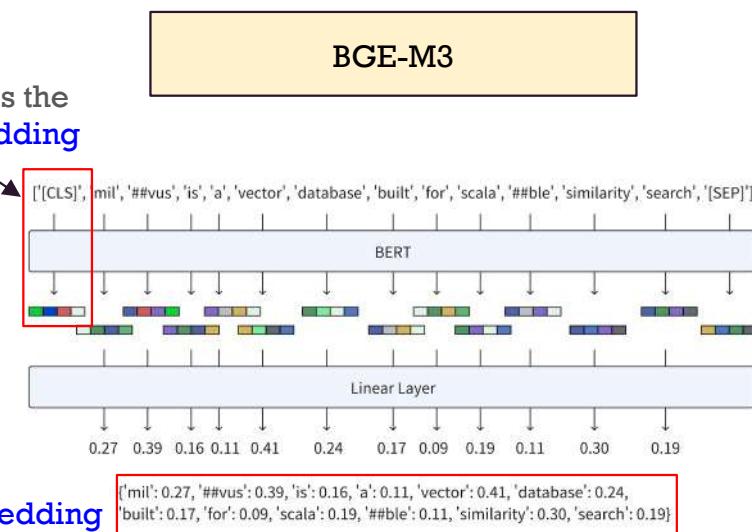
- Basically **contextual search (dense embedding)** + **keyword search (sparse embedding)**
- Supported by many vector DBs.



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

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

[CLS] token  
embedding is the  
**dense embedding**





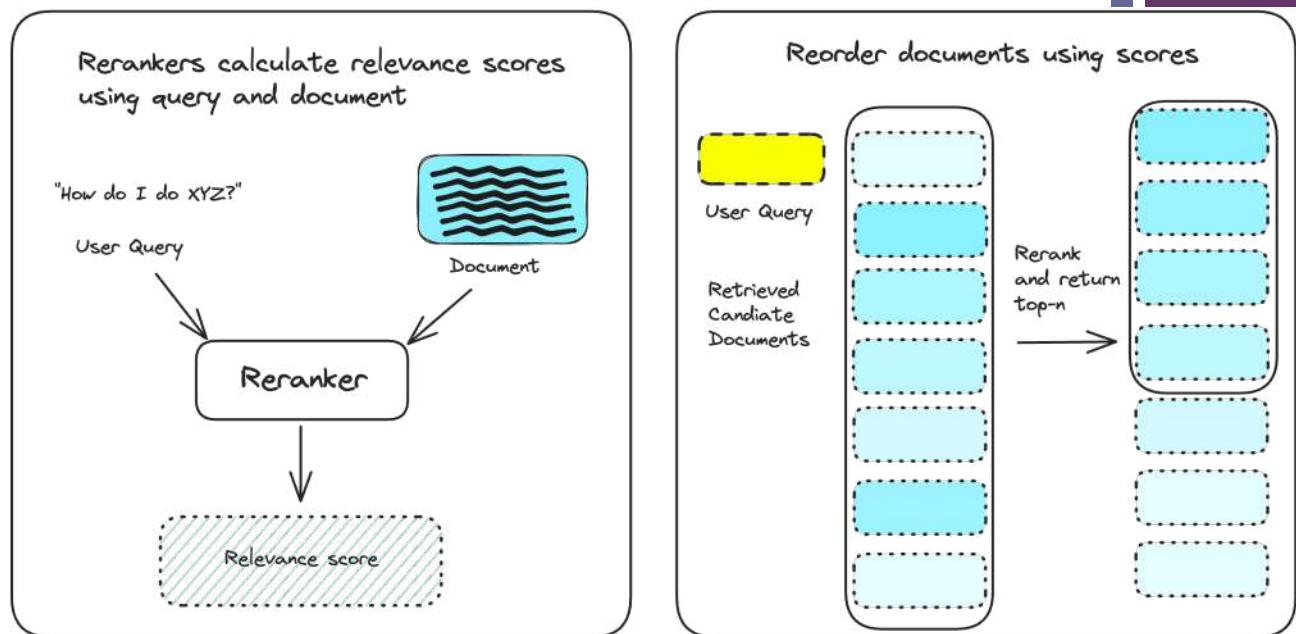
# Document Reranking

- Step3: IR
- 3.1) Use **hybrid search** for better performance
- 3.2) (Optional) - perform **document reranking**

Uses a reranker model or a “cross-encoder” to **rerank the retrieved documents by the relevancy of the query to the documents.**

This process often improves the RAG pipeline performance by weeding out unrelated documents.

Reranker Workflow



Performance on MIRACL dataset

| model                        | inference length | avg          | ar    | bn    | en    | es    | fa    | fi    | fr    | hi    | id    | ja    | ko    | ru    | sw    | te    | th    | zh    | de    | yo    |
|------------------------------|------------------|--------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| bge-m3                       | 512+512          | <b>67.91</b> | 78.4  | 80    | 59.6  | 55.5  | 57.7  | 78.6  | 57.8  | 59.3  | 56    | 72.8  | 69.9  | 70.1  | 78.6  | 86.2  | 82.6  | 61.7  | 56.8  | 60.7  |
| bge-reranker-v2-m3           | 1024             | <b>72.84</b> | 81.7  | 84.63 | 63.45 | 63.71 | 62.49 | 82.41 | 63.26 | 68.25 | 62.71 | 79.96 | 73.79 | 76.93 | 82.27 | 89.36 | 85.3  | 64.2  | 62.64 | 64.03 |
| <b>bge-reranker-v2-gemma</b> | 1024             | <b>73.39</b> | 82.26 | 85.15 | 66.62 | 64.29 | 62.03 | 82.58 | 64.26 | 68.68 | 61.33 | 79.72 | 74.83 | 78.36 | 81.46 | 89.22 | 86.06 | 65.61 | 64.25 | 64.37 |
| bge-reranker-v2-minicpm-20   | 1024             | <b>62.26</b> | 71.8  | 62.15 | 62.51 | 56.45 | 45.28 | 74.73 | 52.77 | 50.64 | 57.3  | 70.57 | 69.12 | 60.58 | 67.4  | 76.98 | 67.87 | 61.61 | 49.74 | 63.2  |
| bge-reranker-v2-minicpm-28   | 1024             | <b>67.75</b> | 77.4  | 64.87 | 66.03 | 61.89 | 51.91 | 80.02 | 63.42 | 56    | 60.5  | 78.25 | 73.39 | 72.5  | 72.09 | 78.41 | 74.46 | 66.09 | 59.15 | 63.07 |
| bge-reranker-v2-minicpm-40   | 1024             | <b>67.77</b> | 77.49 | 64.8  | 66.02 | 62.05 | 51.78 | 80.12 | 62.98 | 55.7  | 60.51 | 78.07 | 73.01 | 72.64 | 72.81 | 78.51 | 74.24 | 65.82 | 59.43 | 63.91 |

<https://huggingface.co/BAAI/bge-m3>

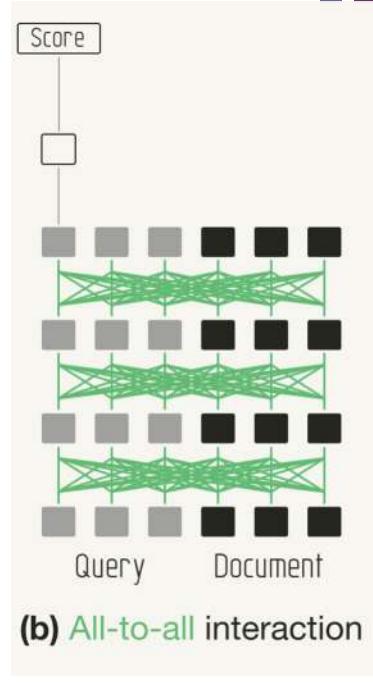
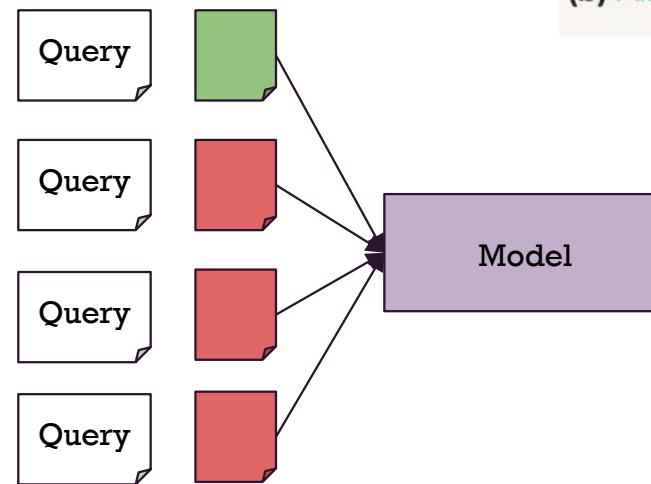


## How to train Reranker (conceptual)

A reranker takes in the query and a document and outputs a score of [0,1] where 1 means the doc is most relevant to the query.

To train a reranker, you will need to feed it:

- A query
- Positive documents
- (Hard) Negative documents
- Labels



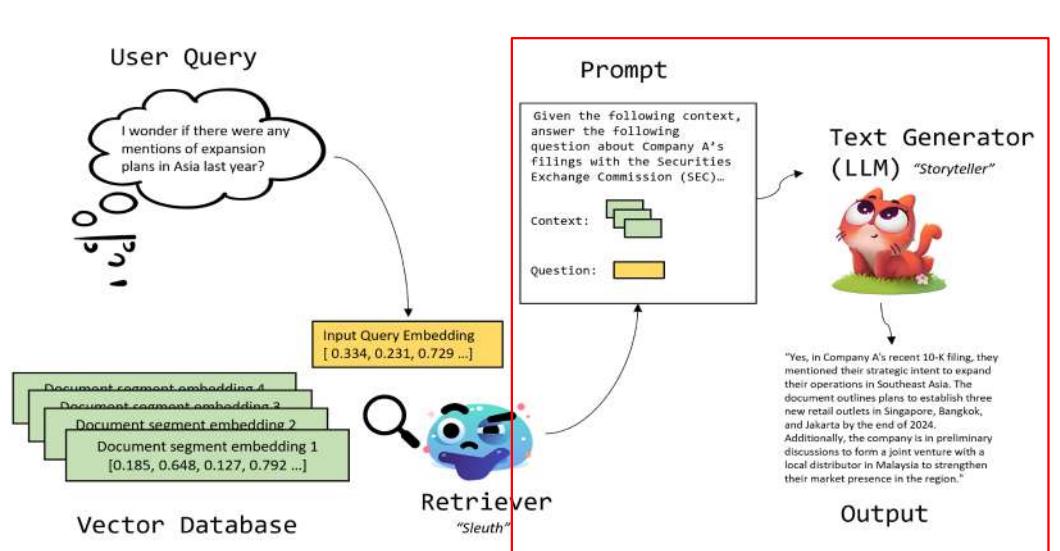
(b) All-to-all interaction

Output: 1,0,0,1

Label: 1,0,0,0

## + Typical RAG Workflow (4)

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





# RAG vs Fine-tuning

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

## Common use cases

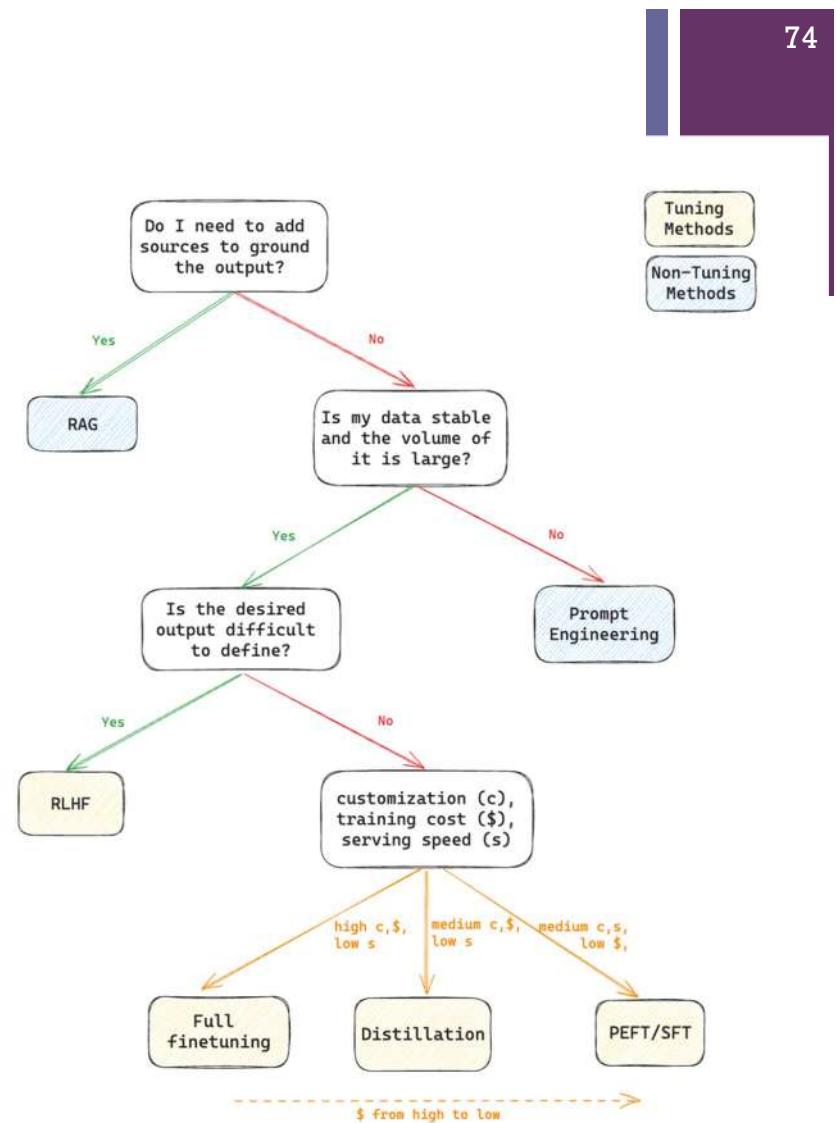
Some common use cases where fine-tuning can improve results:

- 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>



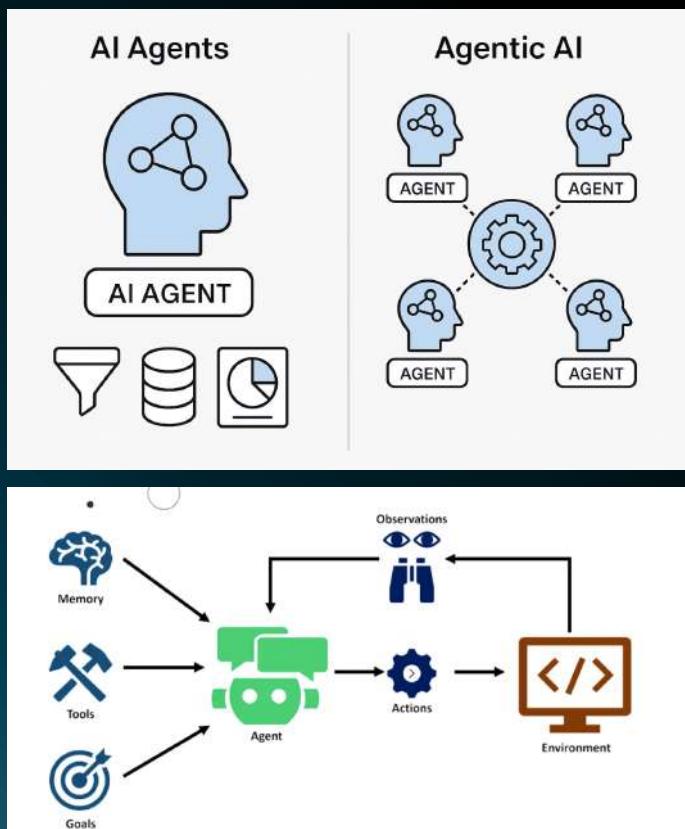
<https://cloud.google.com/blog/products/ai-machine-learning/to-tune-or-not-to-tune-a-guide-to-leveraging-your-data-with-lm>

---

# Agentic LLMs

# From Just Generative AI to **Agentic AI**

**ReAct (Goal) = Reasoning (Thought) + Action + Observe**



Published as a conference paper at ICLR 2023

## REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao<sup>\*,1</sup>, Jeffrey Zhao<sup>2</sup>, Dian Yu<sup>2</sup>, Nan Du<sup>2</sup>, Izhak Shafran<sup>2</sup>, Karthik Narasimhan<sup>1</sup>, Yuan Cao<sup>2</sup>

<sup>1</sup>Department of Computer Science, Princeton University

<sup>2</sup>Google Research, Brain team

<sup>1</sup>{shunyuy, karthikn}@princeton.edu

<sup>2</sup>{jeffreyzhao, dianyu, dunan, izhak, yuancao}@google.com

cs.CL] 10 Mar 2023

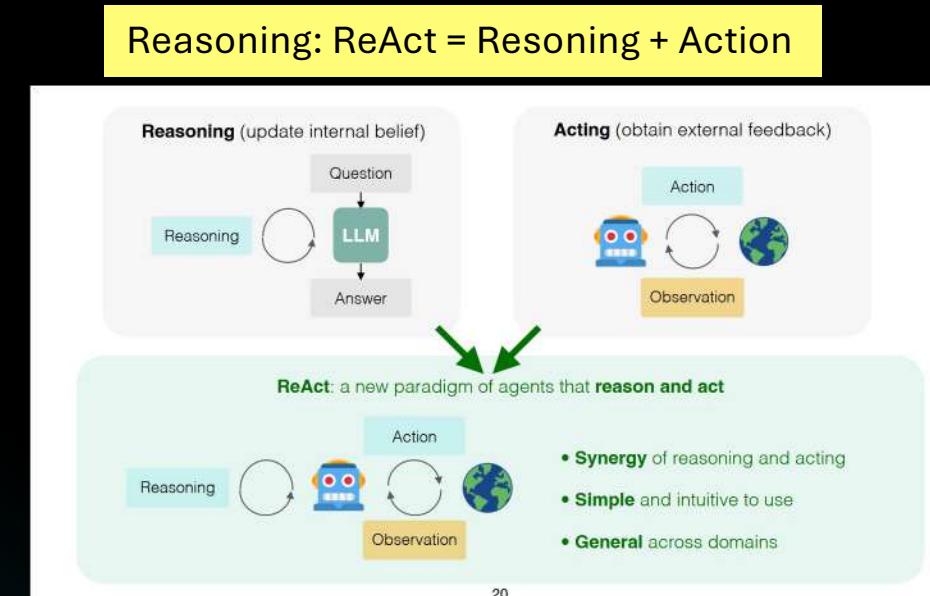
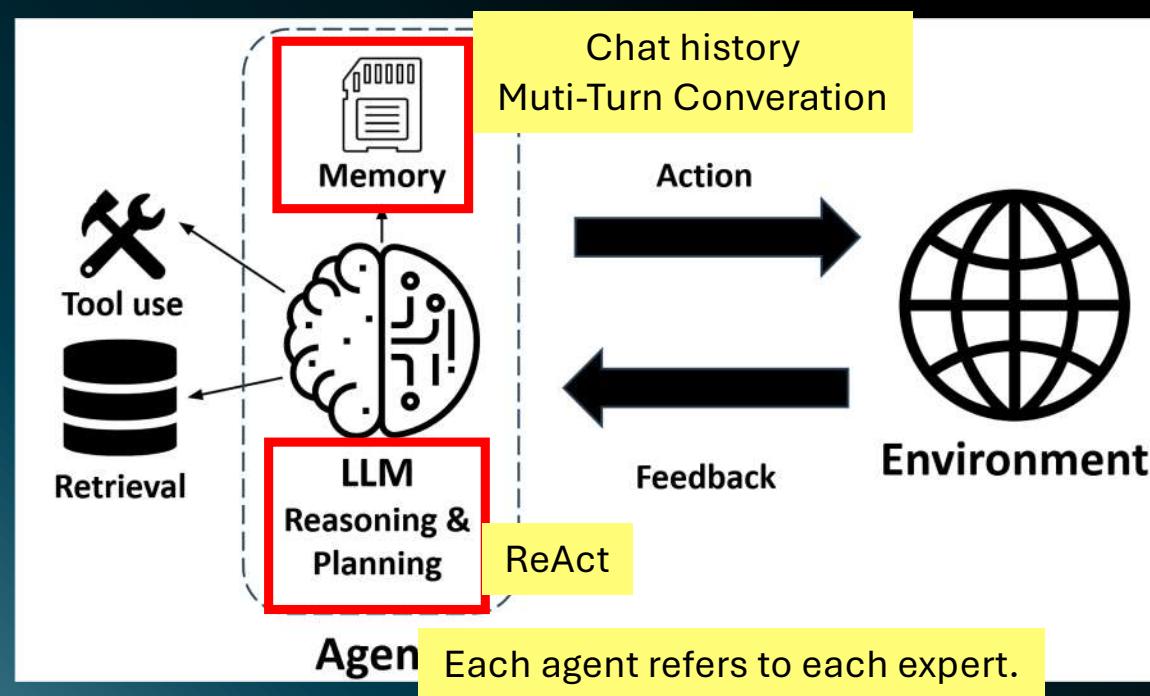
## ABSTRACT

While large language models (LLMs) have demonstrated impressive performance across tasks in language understanding and interactive decision making, their abilities for reasoning (e.g. chain-of-thought prompting) and acting (e.g. action plan generation) have primarily been studied as separate topics. In this paper, we explore the use of LLMs to generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two: reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions allow it to interface with and gather additional information from external sources such as knowledge bases or environments. We apply our approach, named ReAct, to a diverse set of language and decision making tasks and demonstrate its effectiveness over state-of-the-art baselines in addition to

# Agentic LLMs

ReAct = Reasoning (Thought) + Action + Observe

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.



# Single Agents → Multiple Agents



Goal: "Find Q2 2025 sales revenue, compare it to Q2 2024, and report YoY growth."



Different agents = different prompts

## • Step 1



- Thought: I need Q2 2025 sales data first.
- Action: Call search\_docs("Q2 Sales Report 2025")
- Observation: Found document with revenue = \$1.45M

## • Step 2



- Thought: To calculate growth, I also need Q2 2024 data.
- Action: Call search\_docs("Q2 Sales Report 2024")
- Observation: Found document with revenue = \$1.20M

## • Step 3



- Thought: Now I can calculate YoY growth:  $(1.45M - 1.20M) \div 1.20M$
- Action: Call calc\_yoy(current=1.45M, last=1.20M)
- Observation: YoY growth = 20.8%

## • Step 4

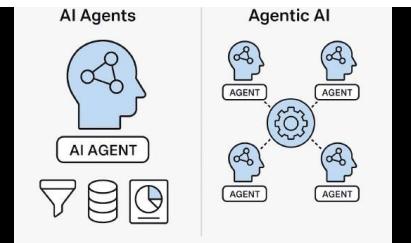


- Thought: I should summarize this result for the CFO.
- Action: Call summarize(revenue\_now=1.45M, revenue\_last=1.20M, yoy=20.8%)
- Observation: "Q2'25 revenue \$1.45M vs Q2'24 \$1.20M → YoY +20.8%"

## • Step 5



- Thought: Finally, I'll draft an email with the summary.
- Action: Call draft\_email(to="cfo@acme.com", subject="Q2 Update", body=summary)
- Observation: Email draft created, ready to send.



# From Just Generative AI to Agentic AI From Chatbot to Personal Assistant

- User: “Book me a round-trip flight from Bangkok to Tokyo next weekend, preferably morning flights, and send me the itinerary.”
- Agentic AI plans, searches, books, and delivers the itinerary automatically.

The screenshot shows a news article from LARA (LaraNews.net) titled "Agentic AI booking tool launched by Kiwi.com". The article is dated August 8, 2025, and is written by Rob Munro. It features a large image of a white airplane with blue accents flying against a dark background. The text discusses the launch of a new AI-powered airline direct booking system by Kiwi.com, which is described as a significant industry disruptor. The article explains that the Czech company is the first to release a Model Context Protocol (MCP) server, which acts as a direct interface between its flight inventory and major AI platforms. The MCP is compared to a "USB-C for AI" due to its standardized communication method.

<https://www.laranews.net/agentic-ai-booking-tool-launched-by-kiwi-com/>

# LLM & Agentic AI Framework

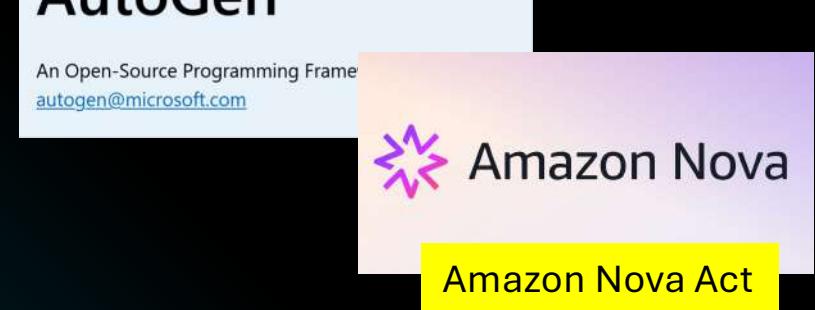
- LLM Framework



- Agentic Framework



An Open-Source Programming Frame  
[autogen@microsoft.com](mailto:autogen@microsoft.com)



Amazon Nova Act

# LLM Tools

# LLM Tools

## LLM Frameworks



microsoft/graphrag

A modular graph-based Retrieval-Augmented Generation (RAG) system



59  
Contributors

227  
Used by

204  
Discussions

21k  
Stars

2k  
Forks



## Agentic Frameworks

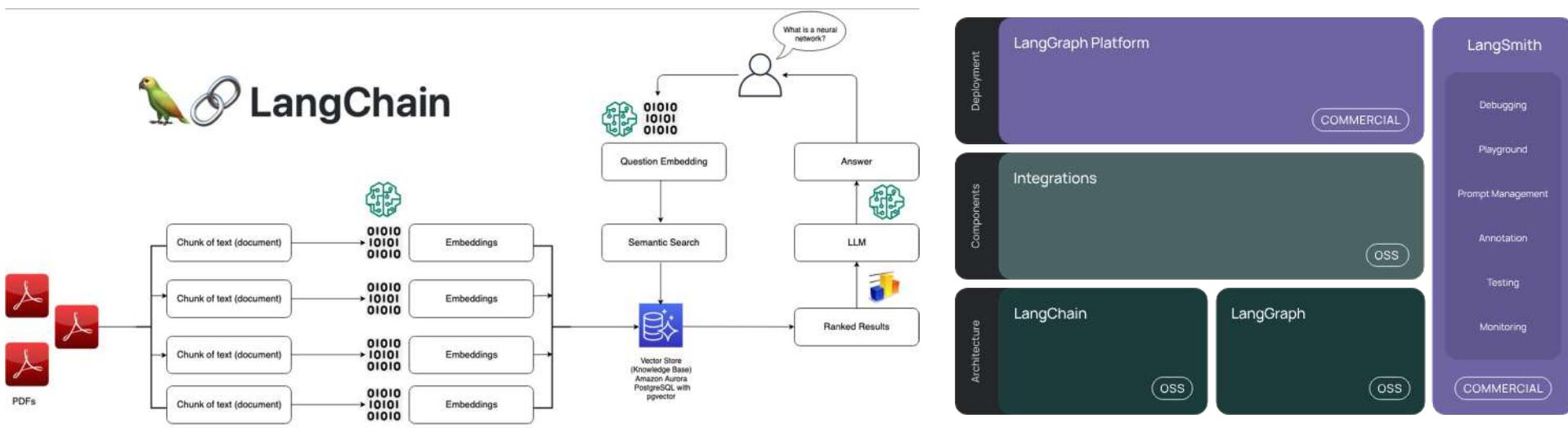


An Open-Source Programming Framework for Agentic AI  
[autogen@microsoft.com](mailto:autogen@microsoft.com)



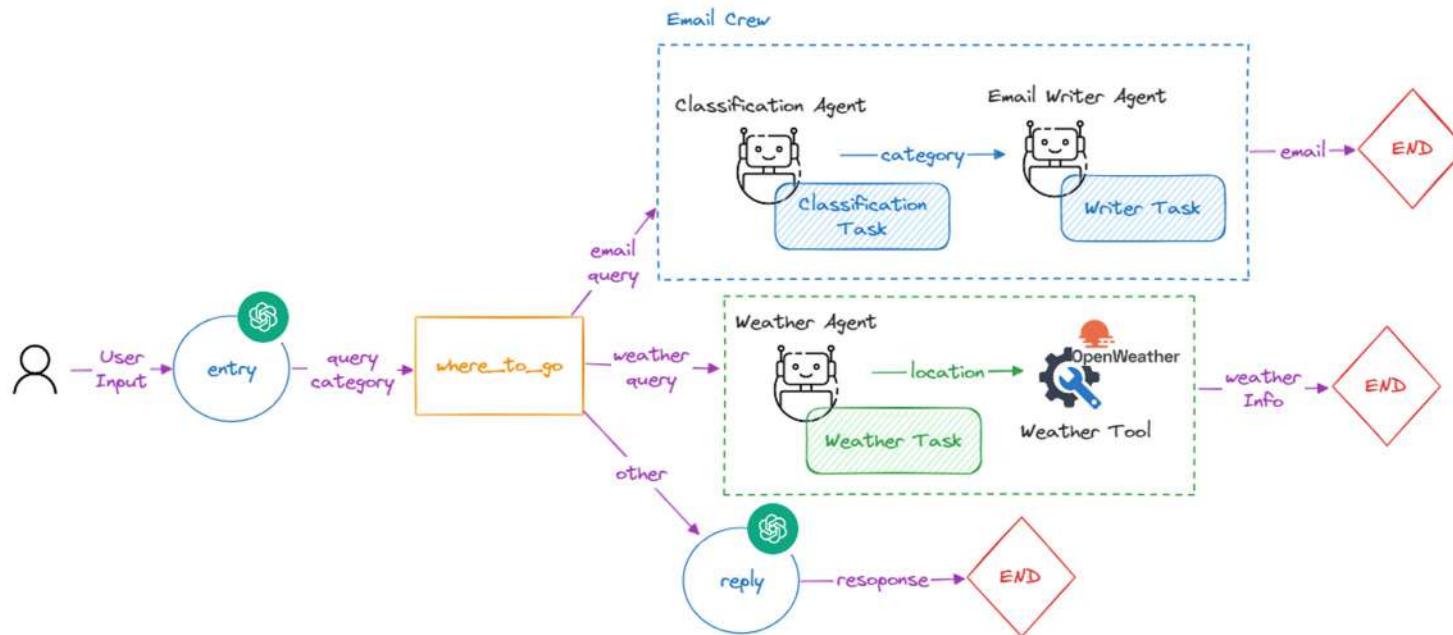
# 1) LangChain

LangChain คือเฟรมเวิร์กที่ช่วยให้นักพัฒนาสร้างแอปพลิเคชันที่เชื่อมต่อและทำงานร่วมกับโมเดลภาษาใหญ่ (LLM) ได้อย่างมีประสิทธิภาพ โดยจัดการการเชื่อมต่อกับข้อมูลภายนอก การควบคุมกระบวนการโต้ตอบหลายขั้นตอน และการเสริมความสามารถของ LLM เช่น การจดจำบริบทและเรียกใช้เครื่องมืออื่น ๆ เพื่อทำงานที่ซับซ้อนได้ดียิ่งขึ้น



## 2) LangGraph

LangGraph is an orchestration framework for complex agentic systems and is more low-level and controllable than LangChain agents.



# 3) LangSmith (LLM Ops)

Langsmith is designed with AI model debugging and orchestration in mind.

## 1) Observability

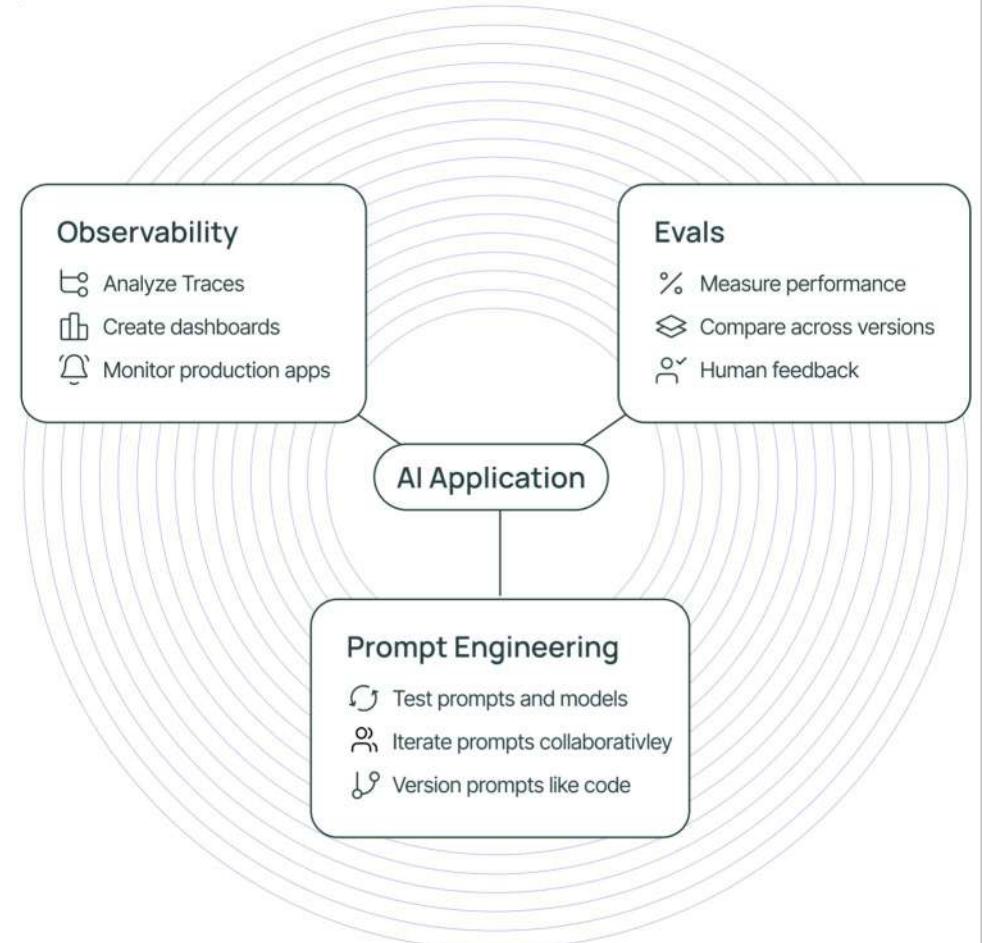
Analyze traces in LangSmith and configure metrics, dashboards, alerts based on these.

## 2) Evals

Evaluate your application over production traffic — score application performance and get human feedback on your data.

## 3) Prompt Engineering

Iterate on prompts, with automatic version control and collaboration features.



---

## Other Topics

- 1) RAG
- 2) Optimization
- 3) Multimodal

## 1

# RAG vs Fine-tuning

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

## Common use cases

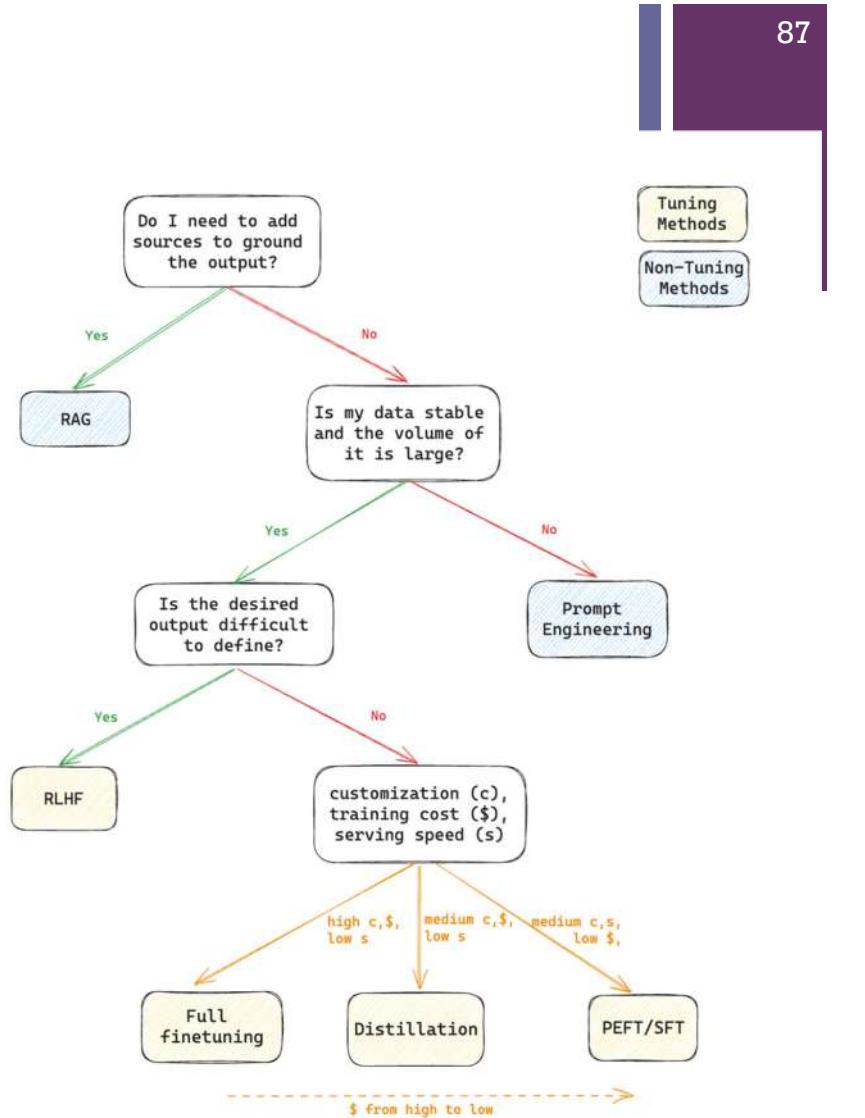
Some common use cases where fine-tuning can improve results:

- 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>



<https://cloud.google.com/blog/products/ai-machine-learning/to-tune-or-not-to-tune-a-guide-to-leveraging-your-data-with-lm>

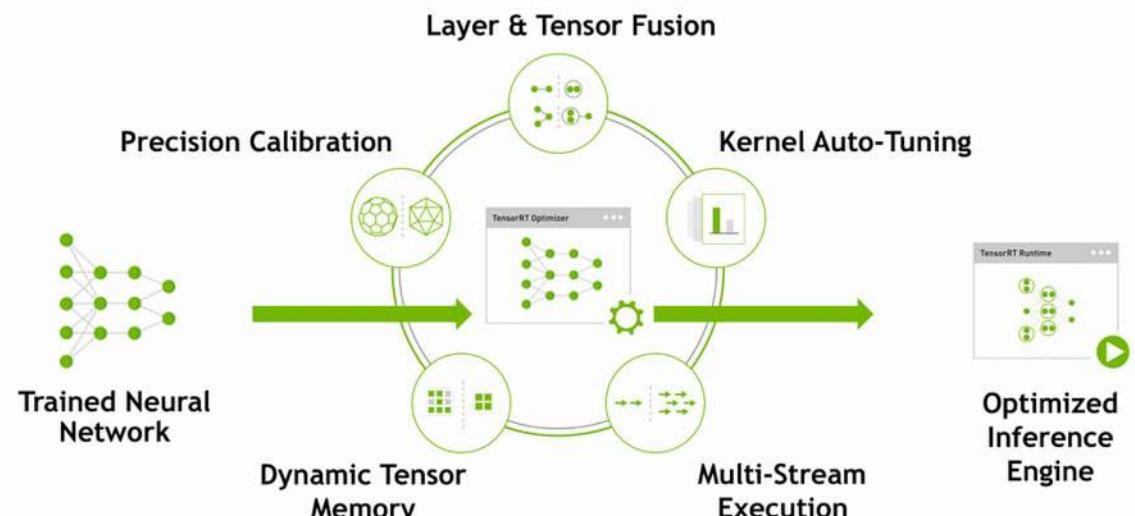


## LLM Optimization is important!

- Only inference
  - 1) TensorRT-LLM + Triton Inference Server
  - 2) NVIDIA Dynamo
- Supporting fine-tuning (training) & inference
  - 3) Unsloth (single GPU - free, but multiple GPUs - paid)
  - 4) DeepSpeed (multiple GPUs - all free, but more complicated than Unsloth)
- Conclusion

## 1) TensorRT: What is it?

TensorRT is a library developed by NVIDIA [for faster inference on NVIDIA GPUs](#). It is built on CUDA, NVIDIA's parallel programming model.



### Features

1. Quantization
2. In-flight Batching
3. Chunked Context
4. KV cache reuse
5. Speculative Sampling

## 2) NVIDIA Dynamo (inference)

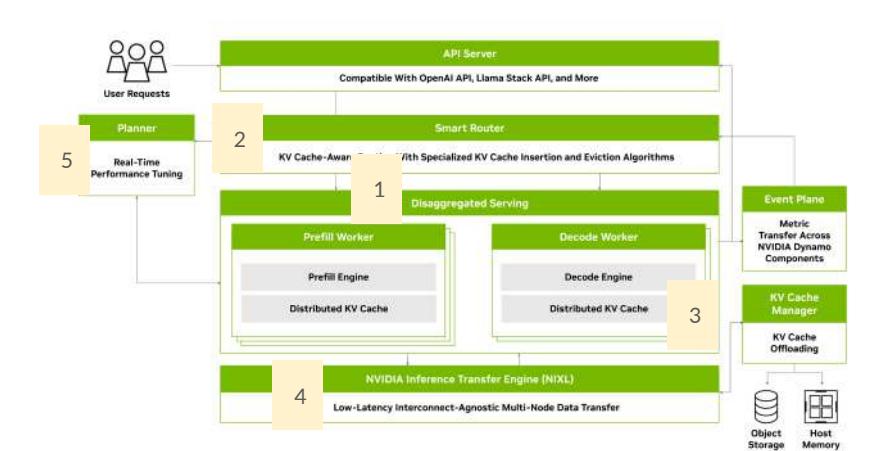
**LLMs: NVIDIA Triton Server → NVIDIA Dynamo**

NVIDIA Dynamo is a high-throughput, low-latency inference framework designed for serving generative AI and reasoning models in **multi-node distributed environments**.

Dynamo is designed to be inference engine agnostic (**supports TRT-LLM, vLLM, SGLang or others**)

To enable large-scale distributed and disaggregated inference serving, Dynamo includes **five key features**.

1. Dynamo Disaggregated Serving
2. Dynamo Smart Router
3. Dynamo Distributed KV Cache Manager
4. NVIDIA Inference Transfer Library (NIXL)
5. Dynamo Planner



### 3) Unsloth: Less VRAM & Larger Batch Sizes

The unsloth library is a [lightweight and high-performance fine-tuning framework for Large Language Models \(LLMs\)](#). It's designed to make parameter-efficient fine-tuning (PEFT) – like LoRA (Low-Rank Adaptation) – fast, simple, and memory-efficient, especially on consumer hardware (like 8GB or 16GB VRAM GPUs).

It builds on top of Hugging Face Transformers + PEFT + BitsAndBytes.

It works by overwriting some parts of the modeling code with optimized operations. By manually deriving backpropagation steps and rewriting all Pytorch modules into Triton kernels.



## Fine-tune Llama 3.3 with Unsloth

Dec 10, 2024 • By Daniel & Michael

Llama 3.3 (70B)

1xA100 80GB

13x  
longer context

Llama 3.3 (70B)

1xA100 80GB

2x  
faster

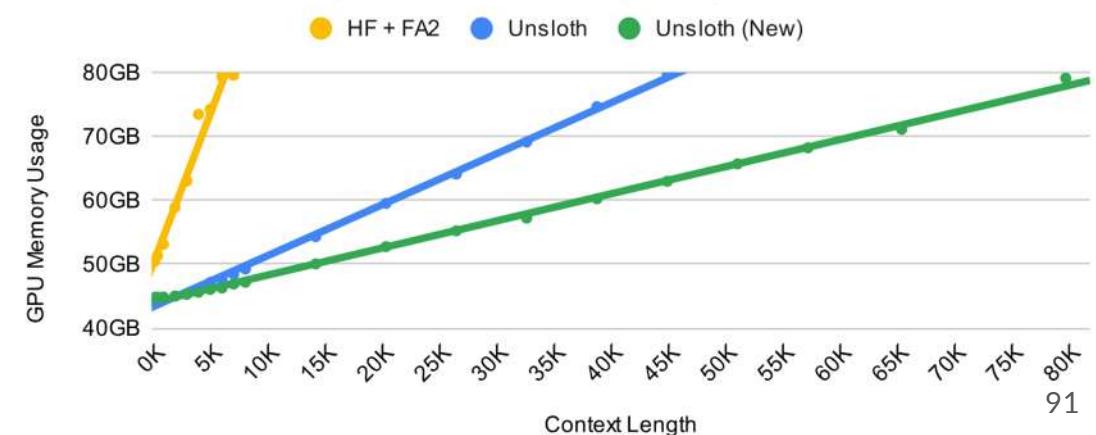
Llama 3.3 (70B)

1xA100 80GB

>75%  
less VRAM

<https://unsloth.ai/>

Llama 3.3 70B Long Context Memory Usage



---

## 4) DeepSpeed: Extreme Speed and Scale for DL Training and Inference

DeepSpeed is an open-source deep learning optimization library developed by [Microsoft](#), designed to train and serve large-scale models efficiently – especially large language models (LLMs) like GPT, BERT, and others. It makes distributed training (multiple GPUs) and inference easy, efficient, and effective.

**DeepSpeed has four innovation pillars:**

### Training

- Speed
- Scale
- Cost
- Democratization
- MoE models
- Long sequence
- RLHF

### Inference

- Large models
- Latency
- Serving cost
- Agility

### Compression

- Model size
- Latency
- Composability
- Runnable on client devices

### Science

- Speed
- Scale
- Capability
- Diversity
- Discovery

## CLIP (Radford et al., 2021)



Learning Transferable Visual Models From Natural Language Supervision

Alec Radford<sup>†</sup> Jong Wook Kim<sup>†</sup> Chris Hallacy<sup>†</sup> Aditya Ramesh<sup>†</sup> Gabriel Goh<sup>†</sup> Sandhini Agarwal<sup>†</sup>  
Girish Sastry<sup>†</sup> Amanda Askell<sup>†</sup> Pamela Mishkin<sup>†</sup> Jack Clark<sup>†</sup> Gretchen Krueger<sup>†</sup> Ilya Sutskever<sup>†</sup>

### Abstract

State-of-the-art computer vision systems are trained to predict a fixed set of predetermined object categories. This restricted form of supervision limits their generality and usability since additional labeled data is needed to specify any other visual concept. Learning directly from raw text about images is a promising alternative which leverages a much broader source of supervision. We demonstrate that the simple pre-training task of predicting which caption goes with which image is an efficient and scalable way to learn a SOTA image representations from scratch on a dataset of 400 million (image, text) pairs collected from the internet. After pre-training, natural language is used to reference learned visual concepts (or describe new ones) enabling zero-shot transfer of the model to downstream tasks. We study the performance of this approach by benchmarking on over 30 different existing computer vision datasets, spanning tasks such as OCR, action recognition in videos, geo-localization, and many types of fine-grained object classification. The model transfers non-trivially to most tasks and is often competitive with a fully supervised baseline without the need for any dataset specific training. For instance, we match the accuracy of the original ResNet-50 on ImageNet million training examples it was trained on. We release our code and pre-trained model weights at <https://github.com/OpenAI/CLIP>.

Task-agnostic objectives such as autoregressive and masked language modeling have scaled across many orders of magnitude in compute, model capacity, and data, steadily improving capabilities. The development of “text-to-text” as a standardized input-output interface (McCann et al., 2018; Radford et al., 2019; Raffel et al., 2019) has enabled task-agnostic architectures to zero-shot transfer to downstream datasets removing the need for specialized output heads or dataset specific customization. Flagship systems like GPT (Brown et al., 2020) are now competitive across many tasks with bespoke models while requiring little to no dataset-specific training data.

These results suggest that the aggregate supervision accessible to modern pre-training methods within web-scale collections of text surpasses that of high-quality crowd-labeled NLP datasets. However, in other fields such as computer vision it is still standard practice to pre-train models on crowd-labeled datasets such as ImageNet (Deng et al., 2009). Could scalable pre-training methods which learn directly from web text result in a similar breakthrough in computer vision? Prior work is encouraging.

Over 20 years ago Mori et al. (1999) explored improving content-based image retrieval by training a model to predict the nouns and adjectives in text documents paired with images. Quattoni et al. (2007) demonstrated it was possible to learn more data efficient image representations via manifold learning in the weight space of classifiers trained to predict words in captions associated with images. Srivastava & Salakhutdinov (2012) explored deep representation learning by training multimodal Deep Boltzmann Machines on top of low-level image and text tag features. Joulin et al. (2016) modernized this line of work and demonstrated that CNNs trained to predict words in image captions learn useful image representations. They converted

<https://arxiv.org/pdf/2103.00020>

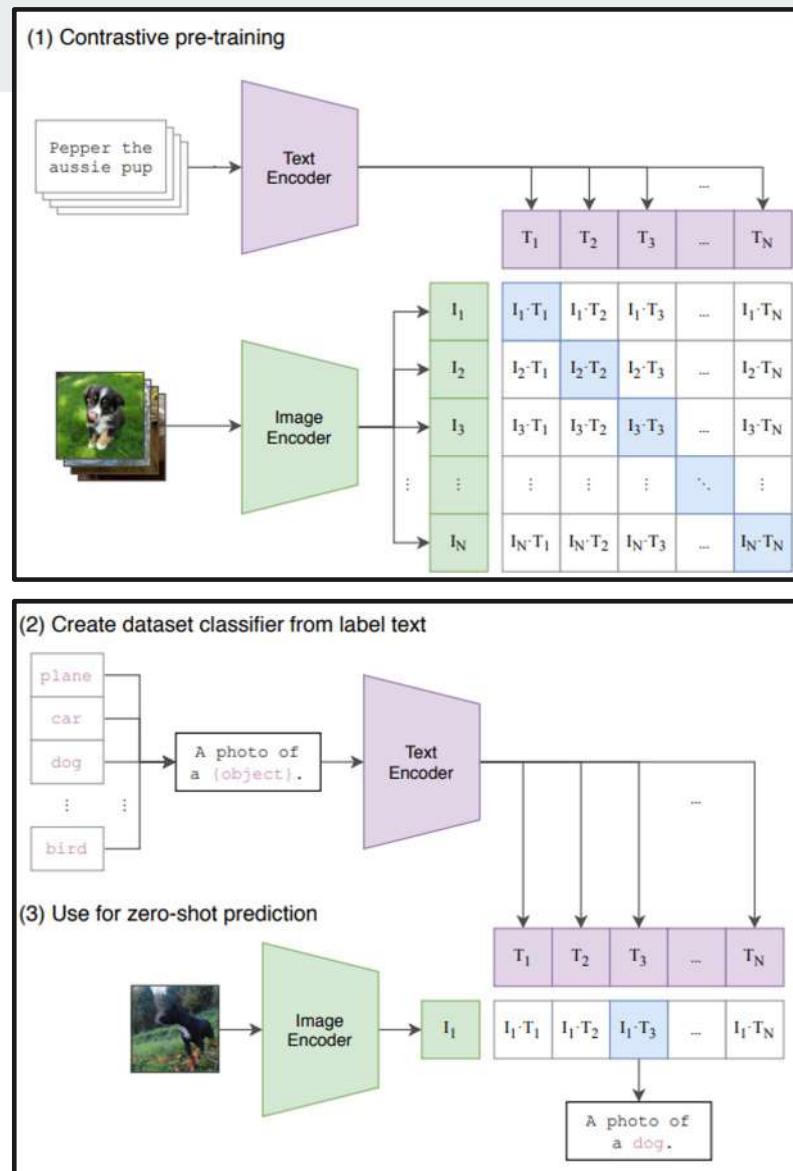
# CLIP (Radford et al., 2021)

CLIP is a model developed by OpenAI that learns joint representations of images and text. It's trained on hundreds of millions of image–text pairs from the internet. By training with contrastive objective called **contrastive learning**.

CLIP is consist of two modules:

1. **Image Encoder:** CLIP leverages both CNN and Vision Transformer (ViT) architectures. For example, ResNet-50 and ResNet-101 are used to transform images into fixed-size embeddings. Alternatively, ViT models such as ViT-B/32, ViT-B/16, and ViT-L/14 can serve as image encoders, converting visual inputs into embedding vectors.
2. **Text Encoder:** A 12-layer Transformer processes tokenized text to generate fixed-size embeddings, enabling direct comparison with image embeddings.

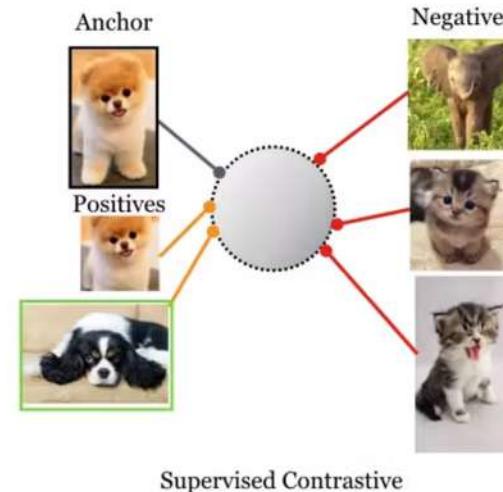
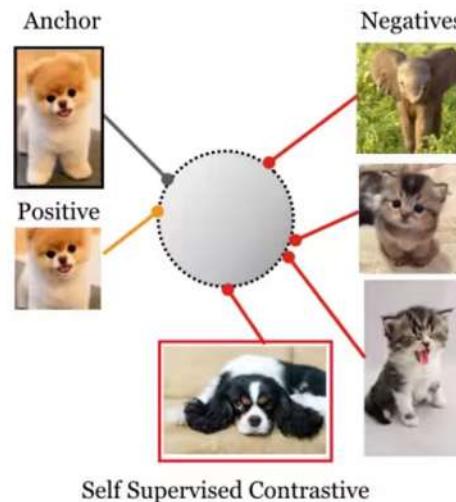
<https://arxiv.org/pdf/2103.00020>



# Contrastive Learning

---

**Contrastive learning** trains a model by aligning matching pairs (e.g., an image of a cat with the text “a cat”) while separating non-matching pairs (e.g., a dog image with “a cat”) in the embedding space. This process teaches the model to link visual and textual concepts, enabling zero-shot transfer to new tasks without task-specific retraining, which make CLIP more flexible and general compared to traditional image models.



# CLIP Huggingface

---

CLIP are available in huggingface published by OpenAI

The screenshot shows the OpenAI profile on Huggingface. The profile header includes the OpenAI logo, 'Enterprise' badge, 'Company' status, 'Verified' checkmark, and a link to <https://openai.com/>. The 'Follow' button has 21,170 followers. The sidebar shows 'AI & ML interests' (None defined yet) and 'Recent Activity' with three entries from tnoah and rahul-openai. The main content area shows search results for 'clip' under 'openai's models'. The results are:

- openai/clip-vit-base-patch32: Zero-Shot Image Classification • Updated Feb 29, 2024 • 15.7M • 761
- openai/clip-vit-base-patch16: Zero-Shot Image Classification • Updated Oct 4, 2022 • 2.26M • 135
- openai/clip-vit-large-patch14: Zero-Shot Image Classification • Updated Sep 1, 2023 • 0.4B • Updated Sep 1, 2023 • 8.16M • 1.86k
- openai/clip-vit-large-patch14-336: Zero-Shot Image Classification • Updated Oct 4, 2022 • 3.95M • 269

Sort: Recently updated

<https://huggingface.co/openai/models?search=clip>

# CLIP Variance

---

There are a lot of variance and many version of improved architecture based on CLIP:

1. **SigLIP** – Improves CLIP's contrastive training by replacing the softmax-based loss with a sigmoid loss for better efficiency and stability.
2. **BLIP / BLIP-2** – Bootstrapping Language–Image Pretraining; extends CLIP with vision–language pretraining for captioning, VQA, and reasoning.
3. **GeoCLIP** – Adapts CLIP for geospatial data and satellite imagery understanding.

---

**Q&A**

**Peerapon.v@chula.ac.th**