Large Language Model (LLM)

2190513 Data Science

Professor Peerapon Vateekul, Ph.D.
peerapon.v@chula.ac.th

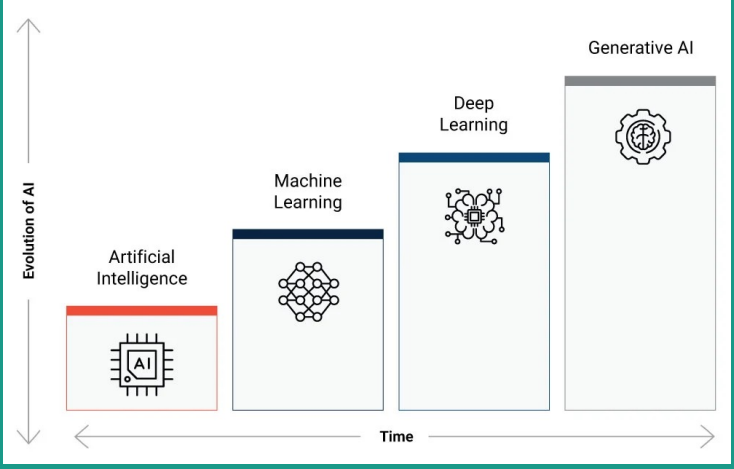
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University

Oultine

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

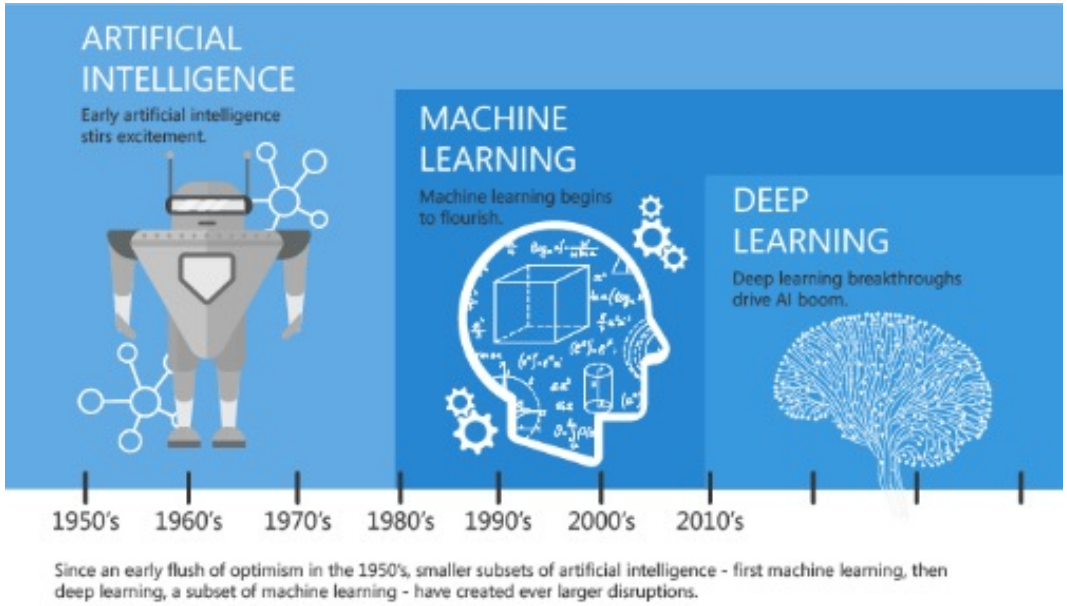
Generative AI: LLM

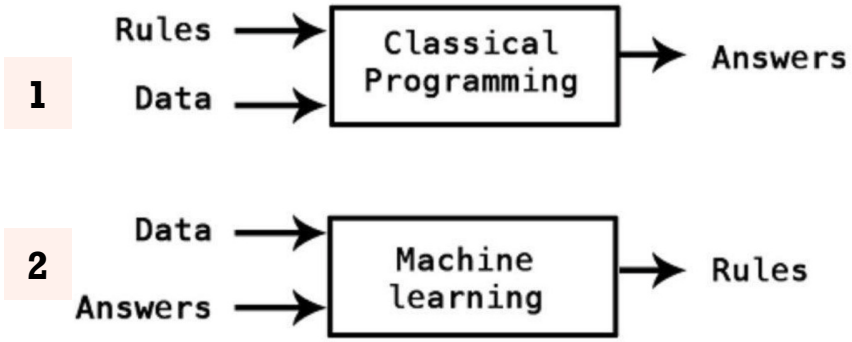
Credit to Asst. Prof.Ekapol's slide

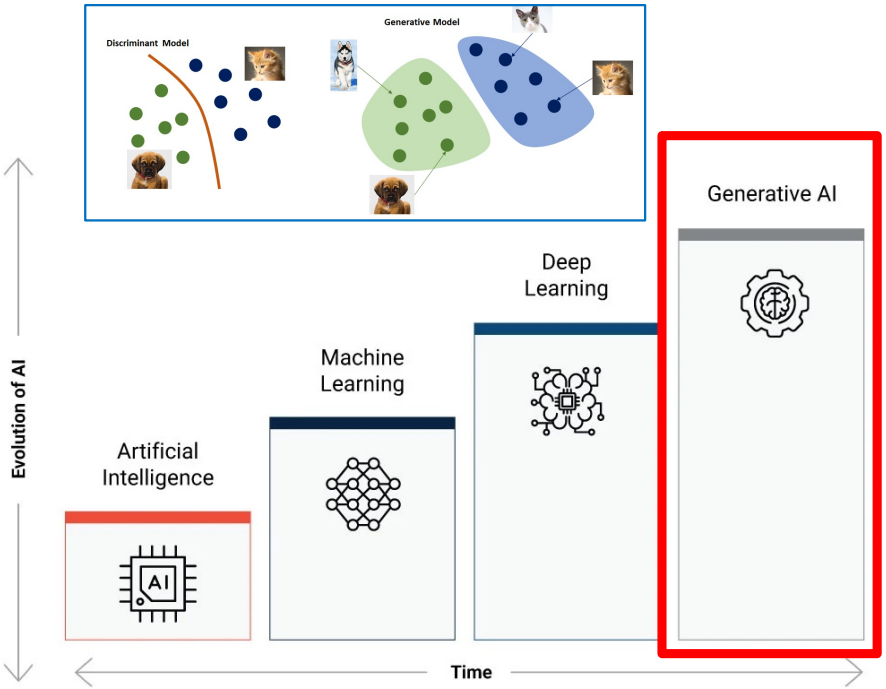


AI = Automation

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

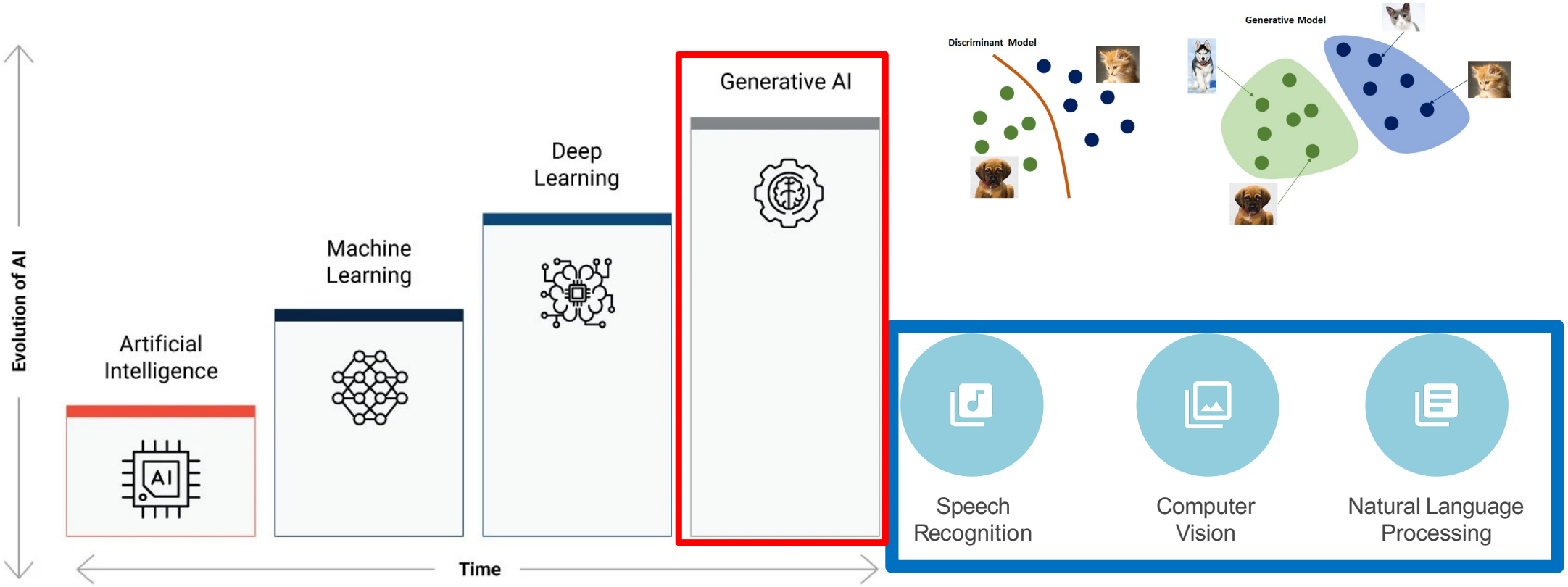




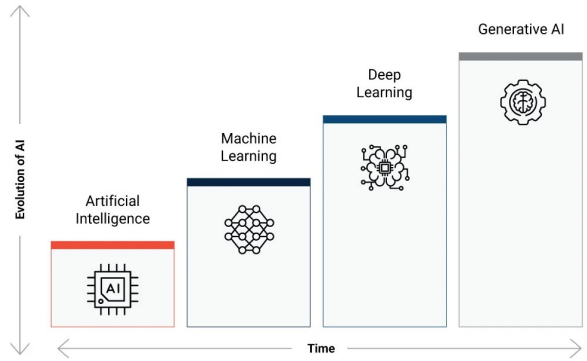


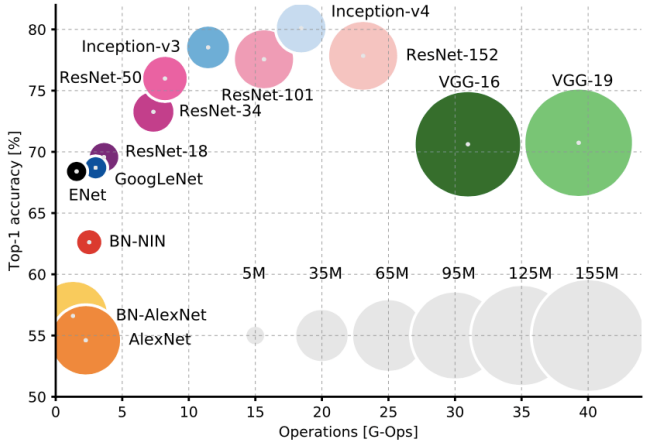
https://mc.ai/machine-learning-basics-artificial-intelligence-machine-learning-and-deep-learning/

Recent Research: DL & Generative AI

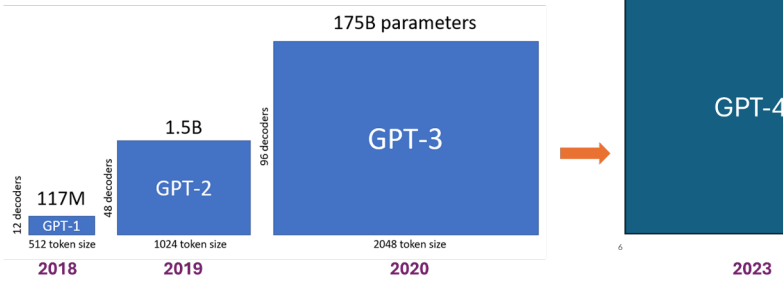


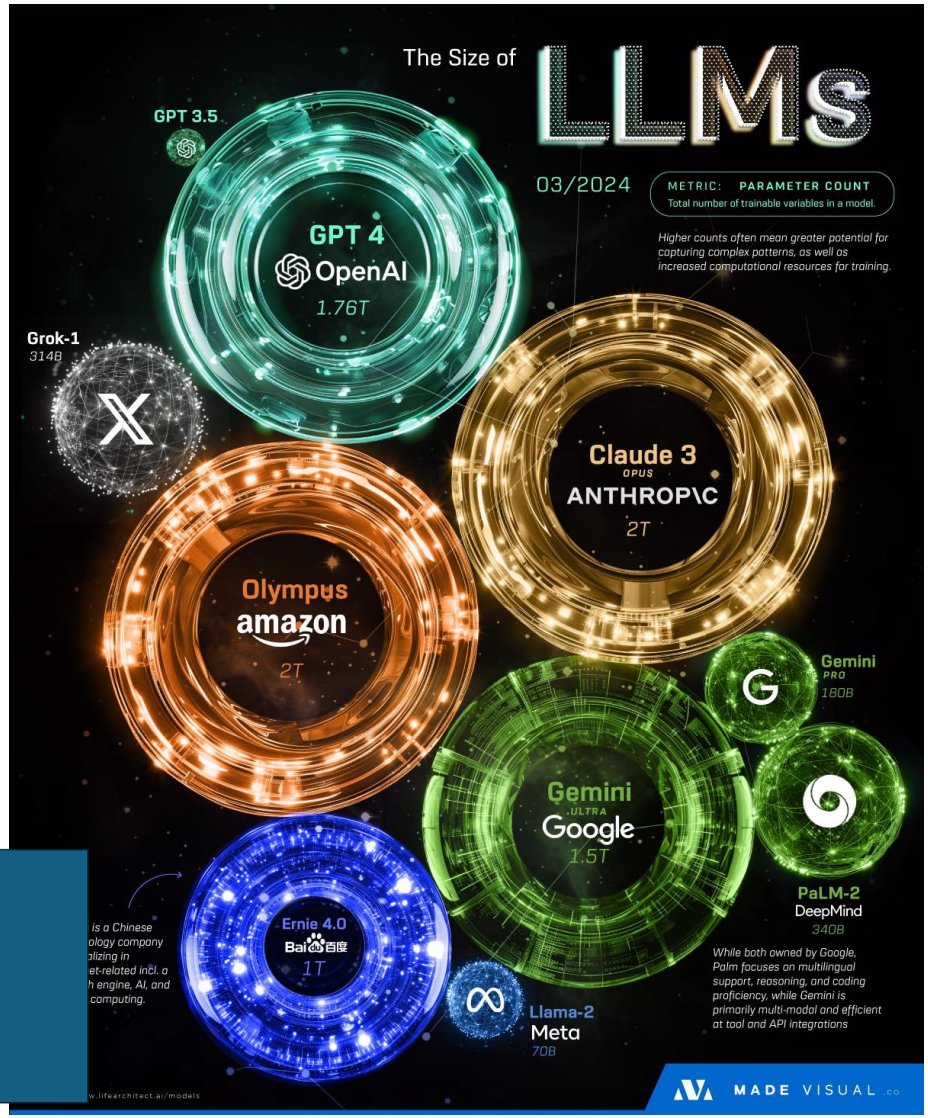
Spend = 500 + 10*Age + 20*Income1K



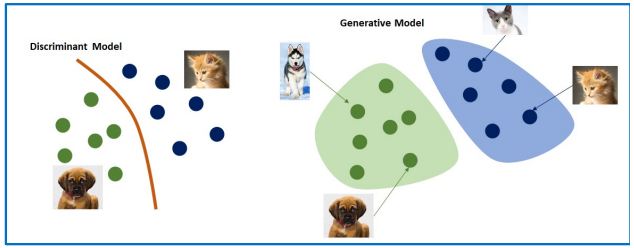


How exactly "Large" it is

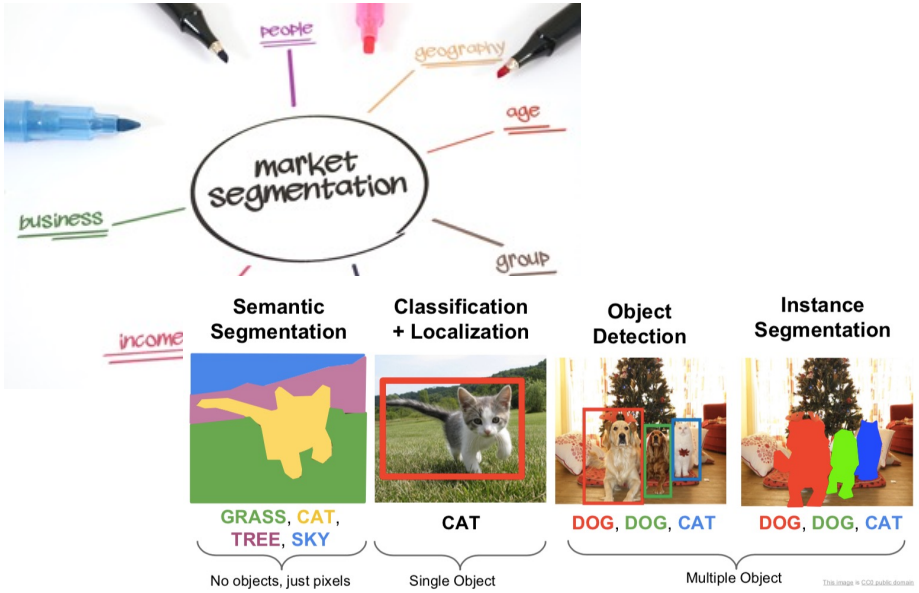




Predictive AI vs. Generative AI



- Predictive AI
  - Predict, Forecast, Detect
  - Cluster, Classify, Segment
- Generative
  - Generate, Generalize



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

AI is looking way too real these days 🤯🍝 #willsmith #ai #sora

Jas Davis
20K subscribers
Subscribe
135

9.2K views 4 months ago
...more

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

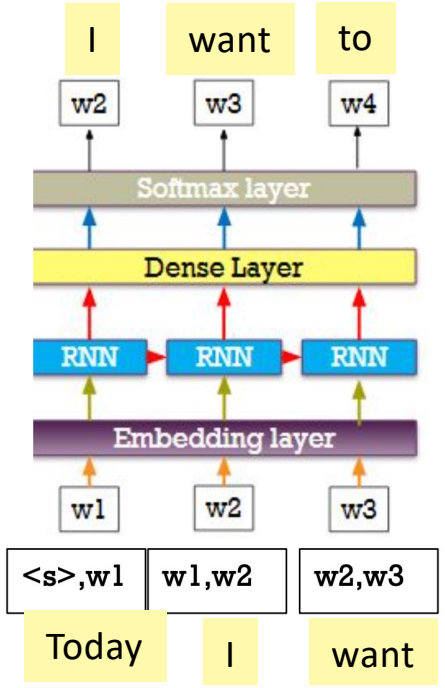
Language Model (LM)

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

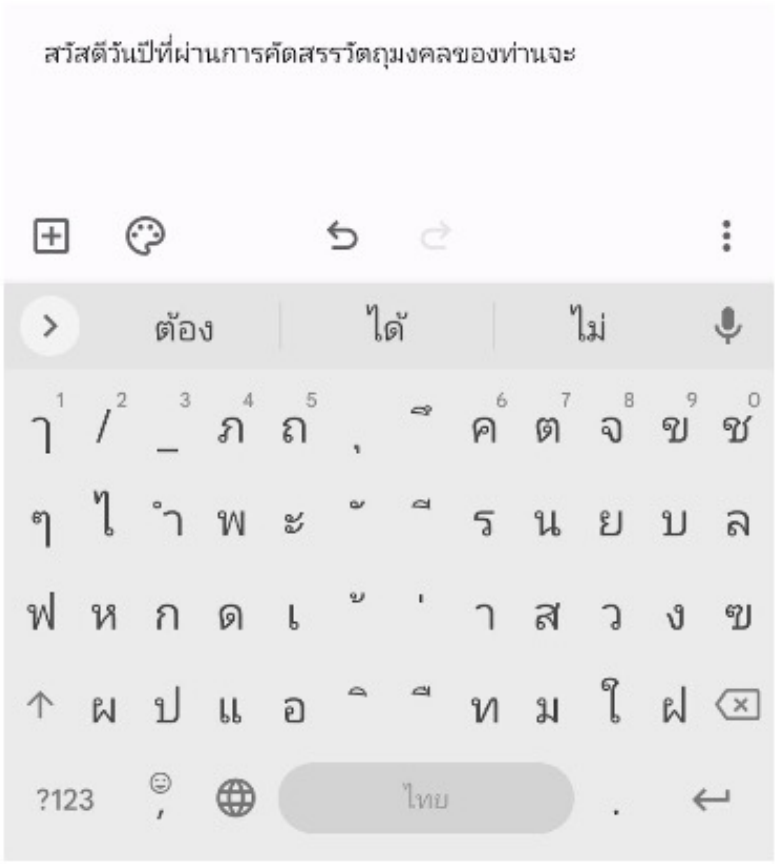
วันนี้เราอยากจะกินข้าวมัน____

Today I want to each chick ____ .

วัน นี้ เรา อยาก จะ กิน ข้าว มัน



You have been using LM ☺ - Autocomplete



LM in 2015

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

SEED: Jobs

Good afternoon. God bless you.

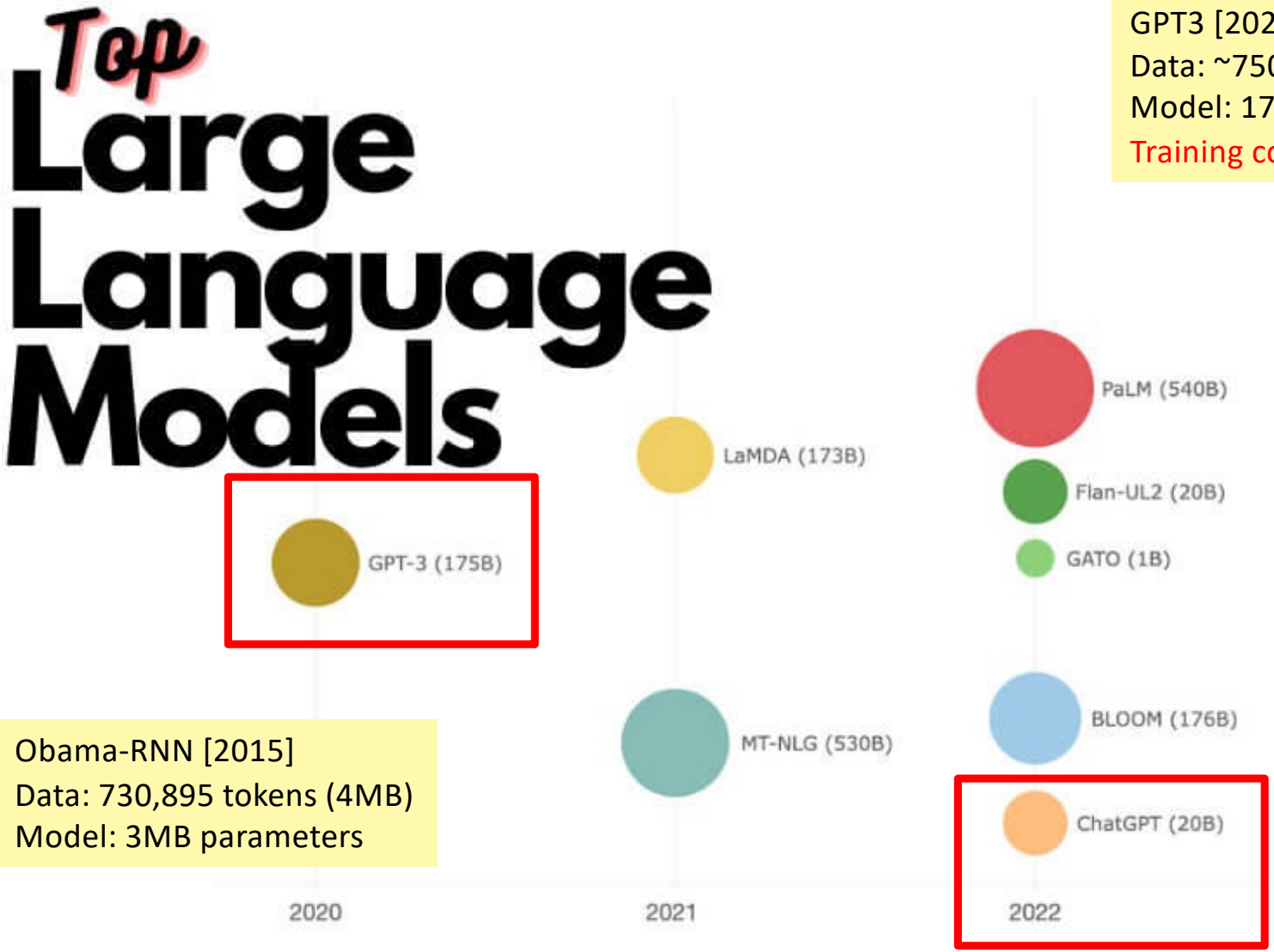
The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done. The promise of the men and women 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 stretcks 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.

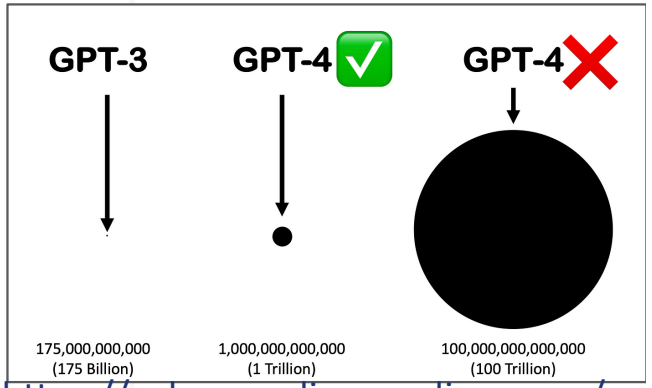
Thank you very much. God bless you, and God bless the United States of America.

https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

Top Large Language Models

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





https://cobusgreyling.medium.com/what-are-realistic-gpt-4-size-expectations-73f00c39b832

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

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

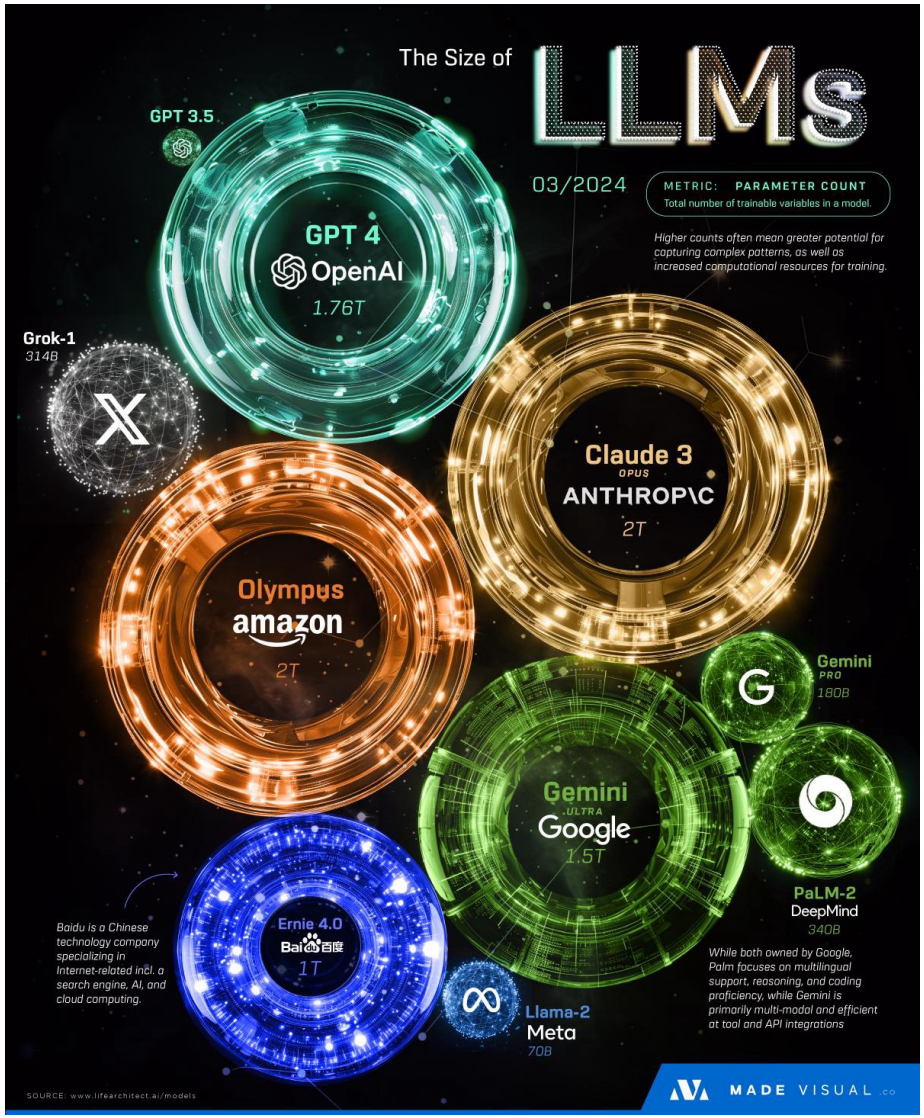
vectara

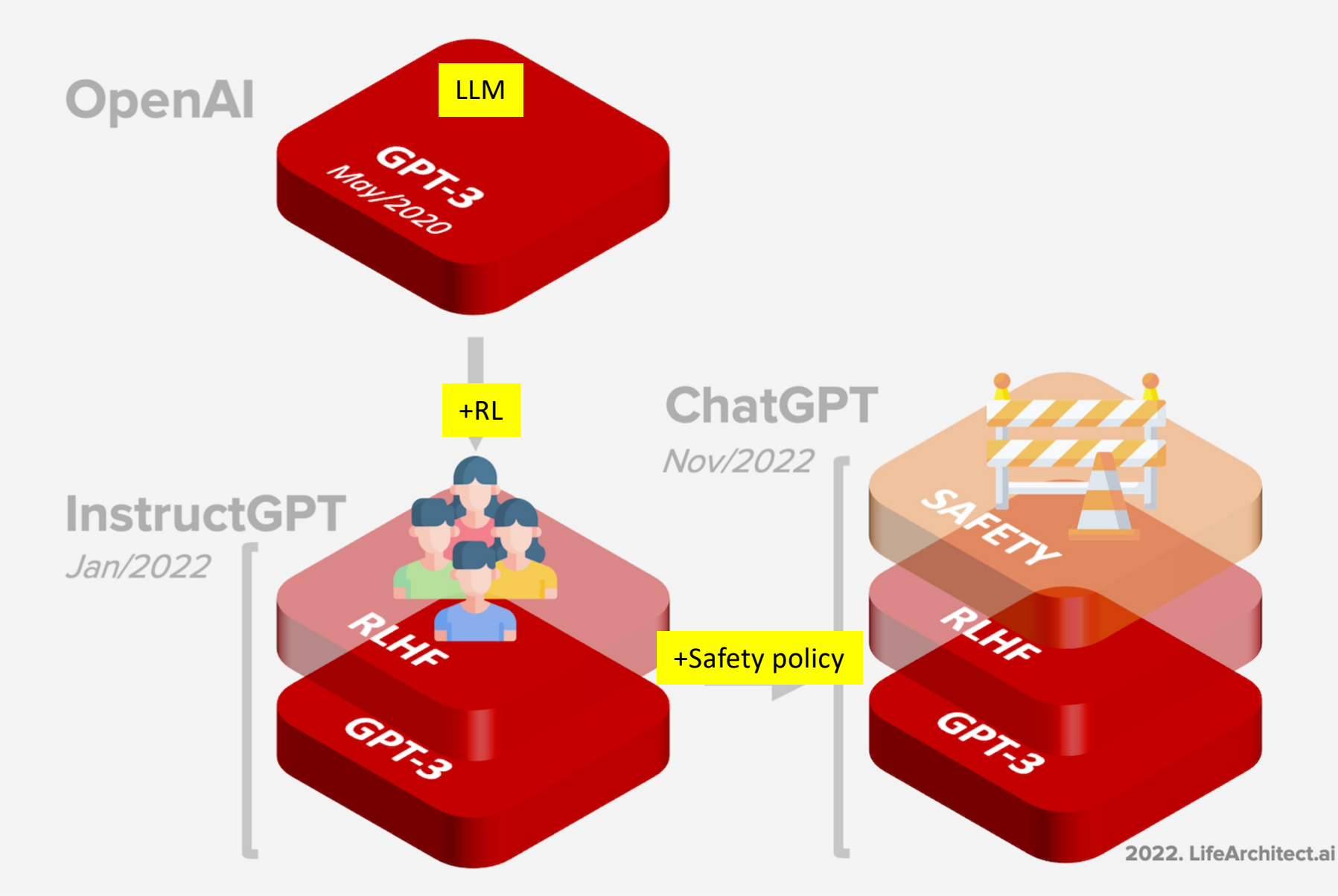
Large + LM = LLM

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
LLM
GPT-3
May/2020
+RL
InstructGPT
Jan/2022
RLHF
GPT-3
+Safety policy
ChatGPT
Nov/2022
SAFETY
RLHF
GPT-3
2022. LifeArchitect.ai

Step 1

Collect demonstration data and train a supervised policy.

LM

A prompt is sampled from our prompt dataset.

Explain reinforcement learning to a 6 year old.

A labeler demonstrates the desired output behavior.

We give treats and punishments to teach...

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

SFT

Step 2

Collect comparison data and train a reward model.

Reward Model (Scoring Model)

A prompt and several model outputs are sampled.

Explain reinforcement learning to a 6 year old.

A: In reinforcement learning, the agent is...

B: Explain rewards...

C: In machine learning...

D: We give treats and punishments to teach...

A labeler ranks the outputs from best to worst.

D > C > A > B

This data is used to train our reward model.

RM

D > C > A > B

Step 3

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

RL

A new prompt is sampled from the dataset.

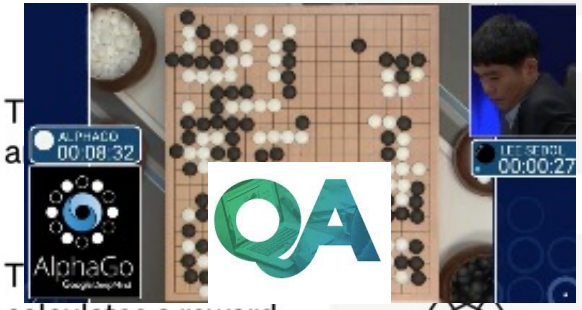
Write a story about otters.

The PPO model is initialized from the supervised policy.

PPO

T a

T calculates a reward for the output.



The reward is used to update the policy using PPO.

r_k

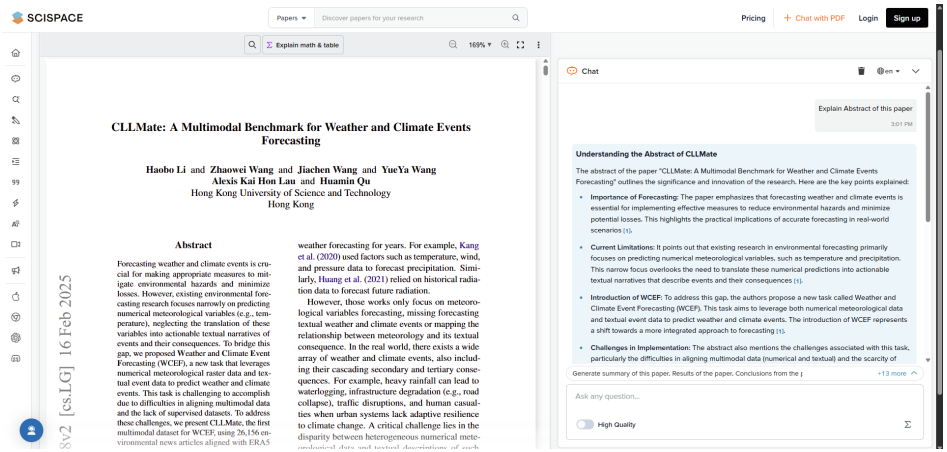
https://www.hpc-ai.tech/blog/colossal-ai-chatgpt

LLM Capabilities

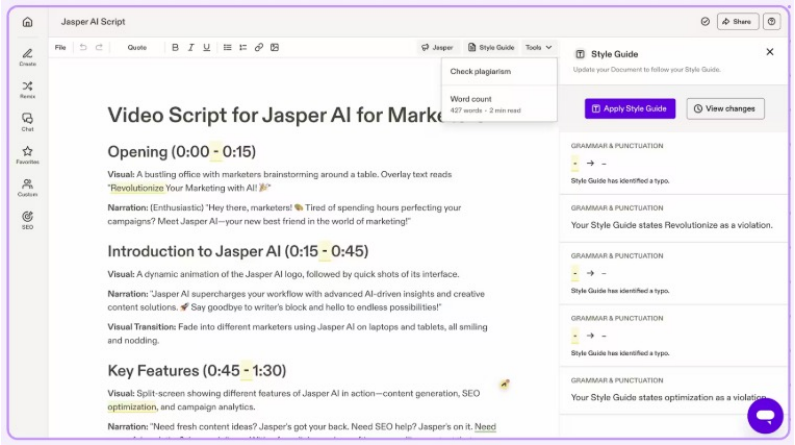
ChatGPT

1) Assistant Chatbot

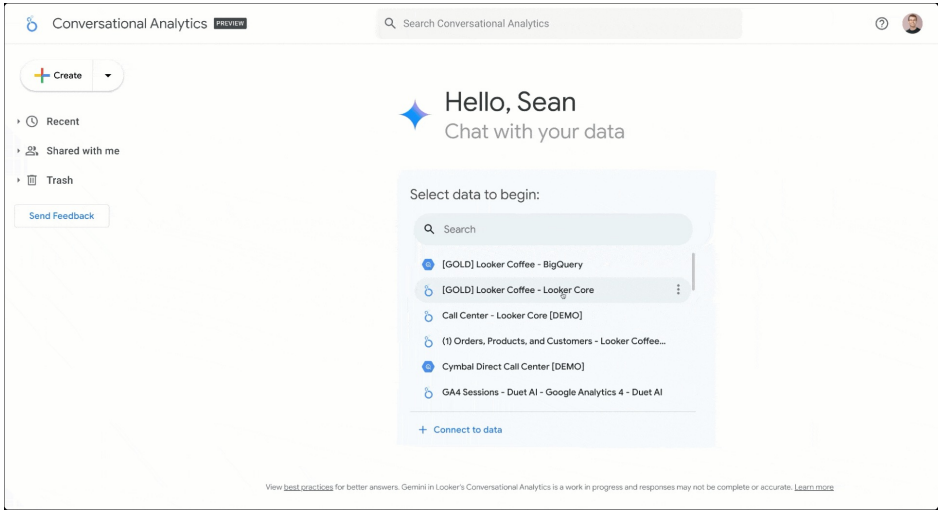
2) Text Summarization



3) Content Creation

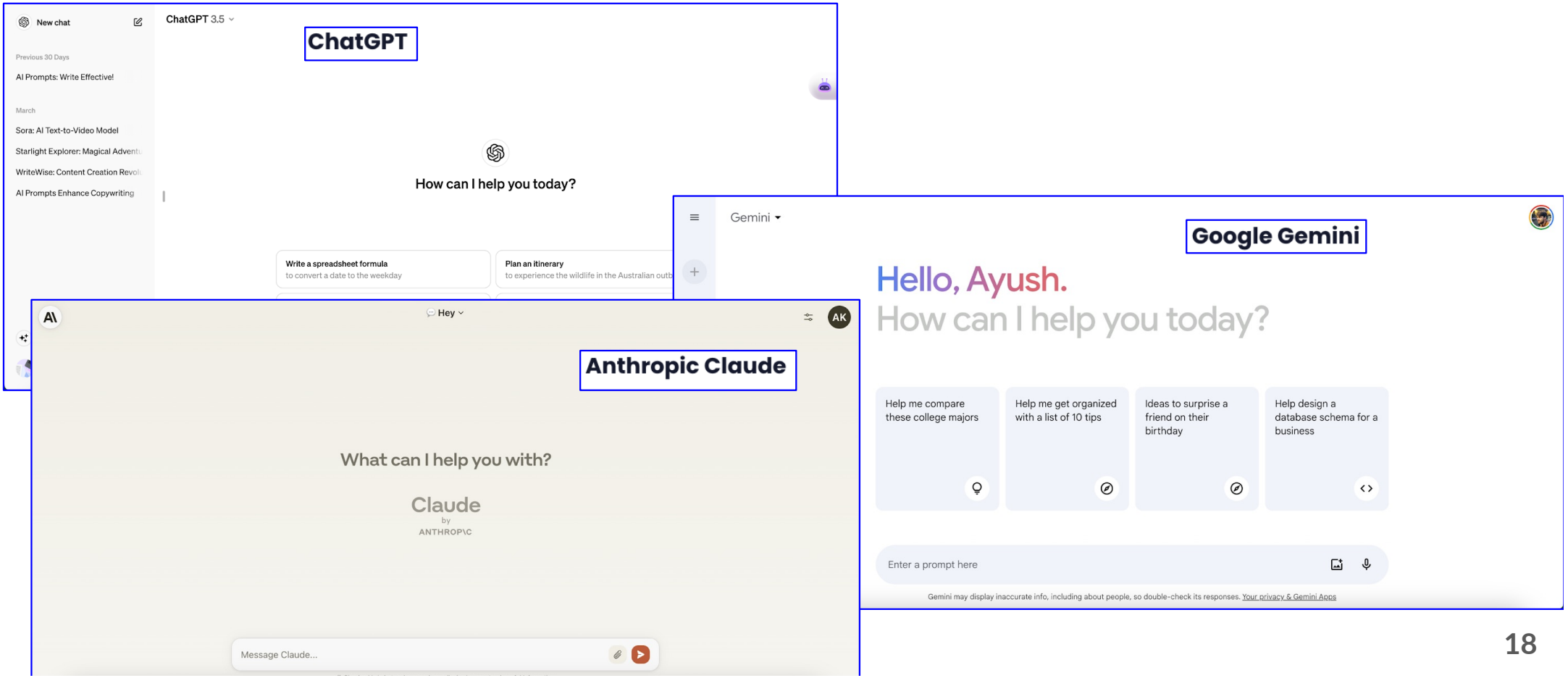


4) Data Analytics



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

AI text generation tools



Modern LLMs in 2025 – beyond textual data

Category	Capabilities	Examples
1) Omni Models	Multimodal + tools + real time	GPT-o, Gemini 2.0, Claude 3.7
2) Reasoning Models	Math, logic, planning	o1/o3, DeepSeek-R1, Gemini Thinking
3) Vision-Language Models	OCR, charts, grounding	Qwen2-VL, InternVL, GPT-o Vision
4) Code Models	Program synthesis, debugging	o1-coder, Gemini Code, DeepSeek-Coder
5) Agentic Models	Tool use, autonomy	OpenAI Agent, AutoGPT-X, Meta Agent
6) Long Context	1–10M tokens	Gemini 2.0, Claude 3.7, Llama3.3
7) Voice AI	Real-time speech AI	GPT-o Voice, Gemini Live

Commercial Large Language Model (LLM)

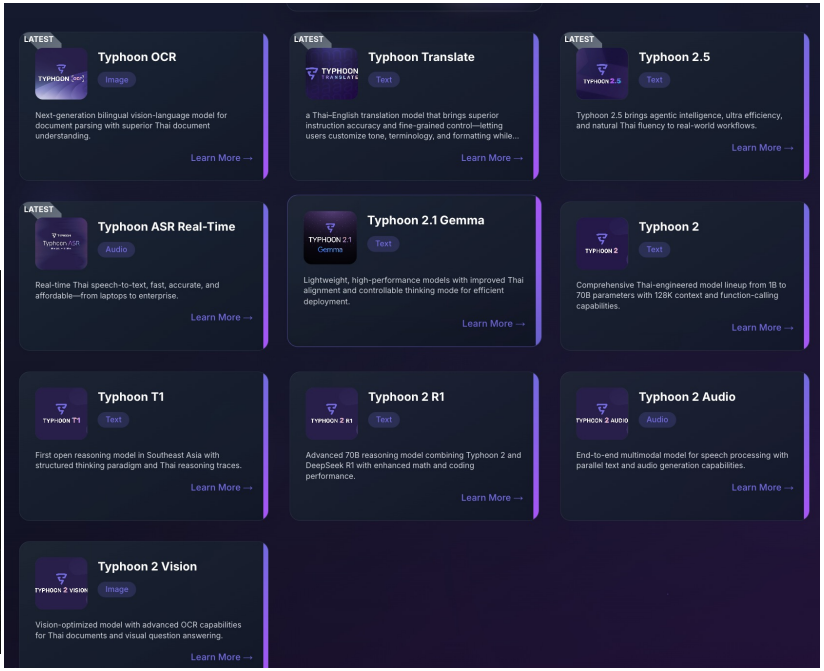
Global LLMs: OpenAI GPT, Google Gemini, LLaMA etc.

OpenAI ChatGPT

Gemini

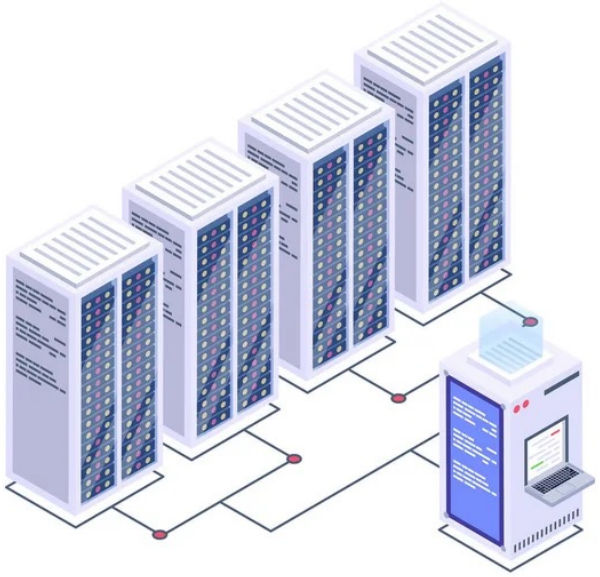
LLaMA by Meta

Thai LLMs: OpenThaiGPT, Typhoon, etc.



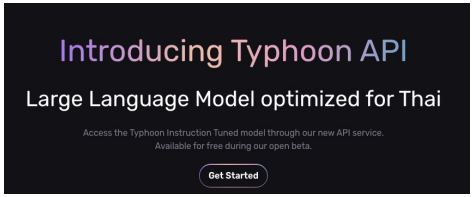
(1) Loading a Pretrained Model Locally vs. (2) Using a Cloud-based API LLM

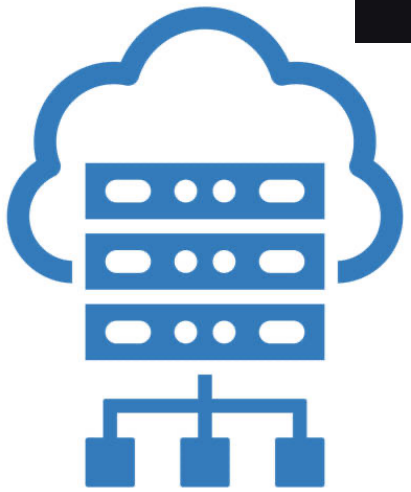
LLaMA by Meta



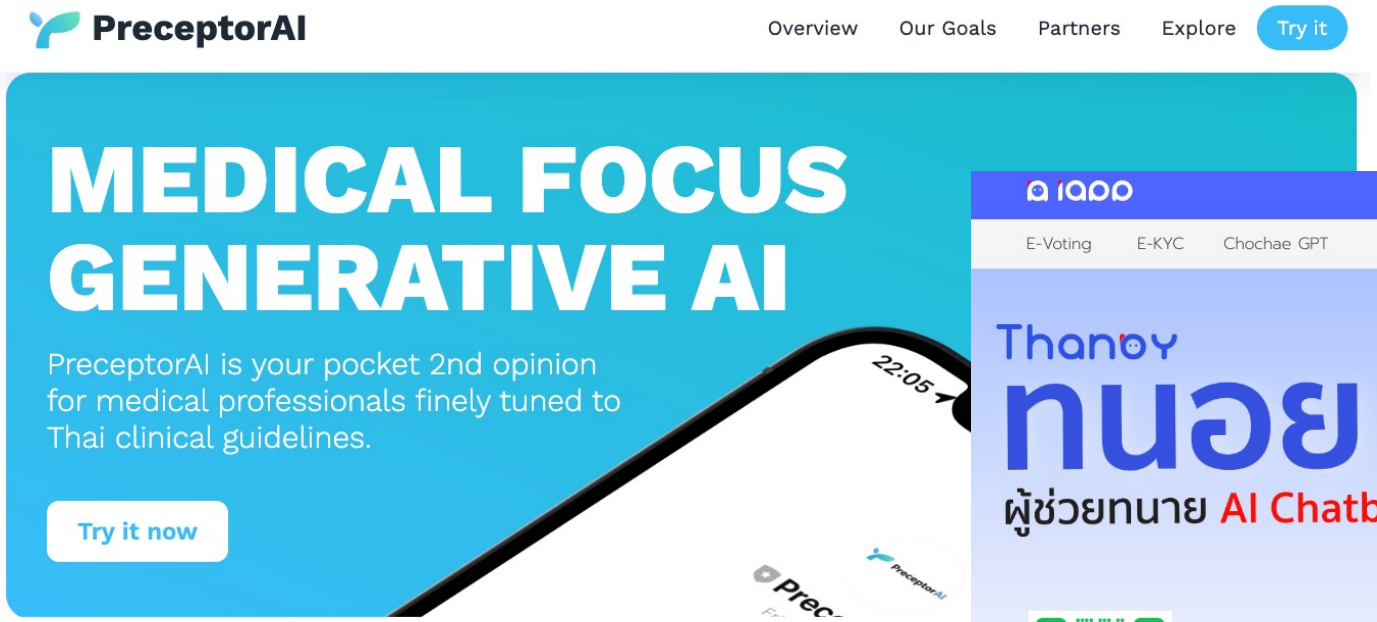
OpenAI ChatGPT

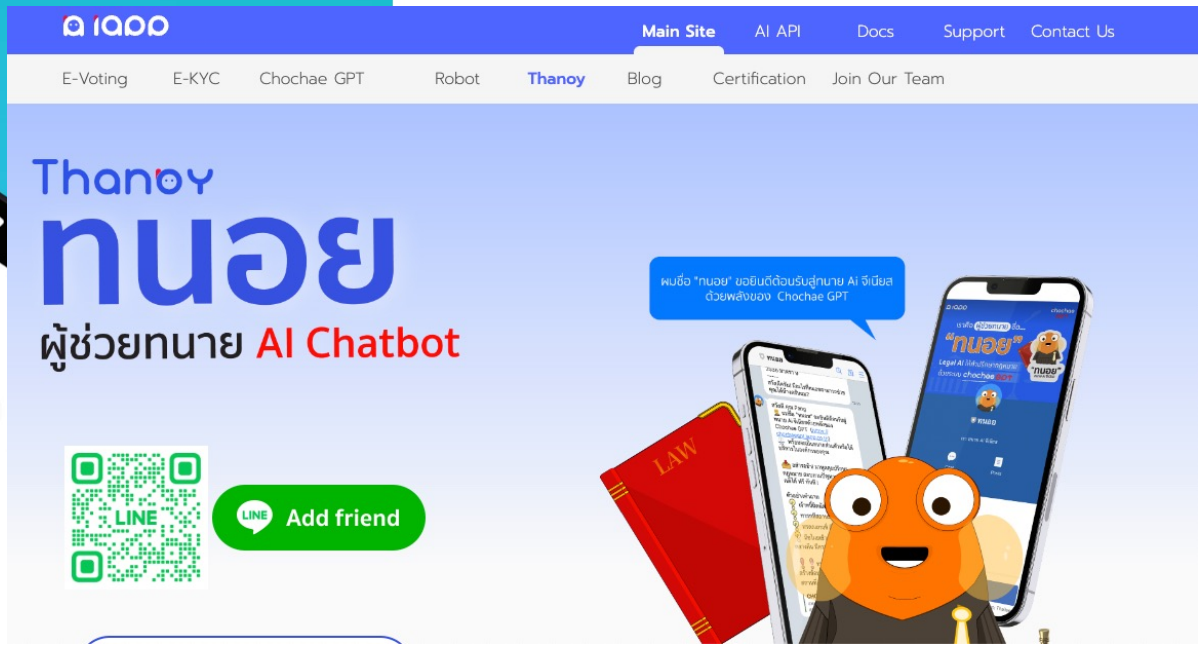
Gemini





LLM Use Cases: Domain Specific Chatbot



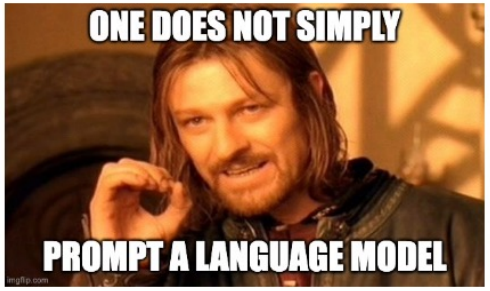


Prompting

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

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

Prompt Engineering Techniques

servicenow

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

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.

```
const response = await openai.chat.completions.create({
  model: "gpt-4o",
  messages: [
    {
      "role": "developer",
      "content": [
        {
          "type": "text",
          "text": `
            You are a helpful assistant that answers programming
            questions in the style of a southern belle from the
            southeast United States.
          `
        }
      ]
    },
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": "Are semicolons optional in JavaScript?"
        }
      ]
    }
  ],
  store: true,
});
```

```
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.", # <-- rol
    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"

https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/

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)
```

```
Providing clear and specific instructions to a model is essential for gui
ding 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 bright
est 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 fet
ch water from a hilltop well, but encounter misfortune along the way.

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

3 - Jack, Jill

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

- 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 + 100x
Total cost: 100x + 250x + 100,000 + 100x = 450x + 100,000
"""
response = get_completion(prompt)
print(response)
```

```
The student's solution is correct. The total cost for the first year of o
perations as a function of the number of square feet is indeed 450x + 10
0,000.
```

With "thinking"

Answer correctly as "False"

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

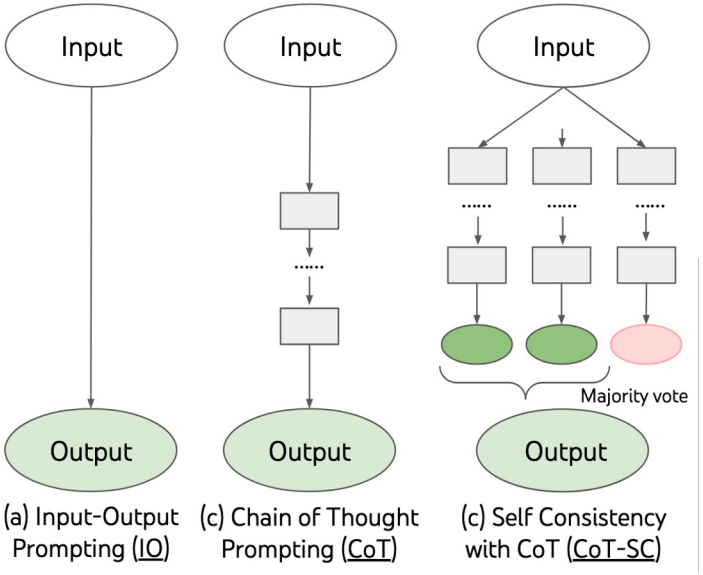
We can fix this by instructing the model to work out its own solution first.

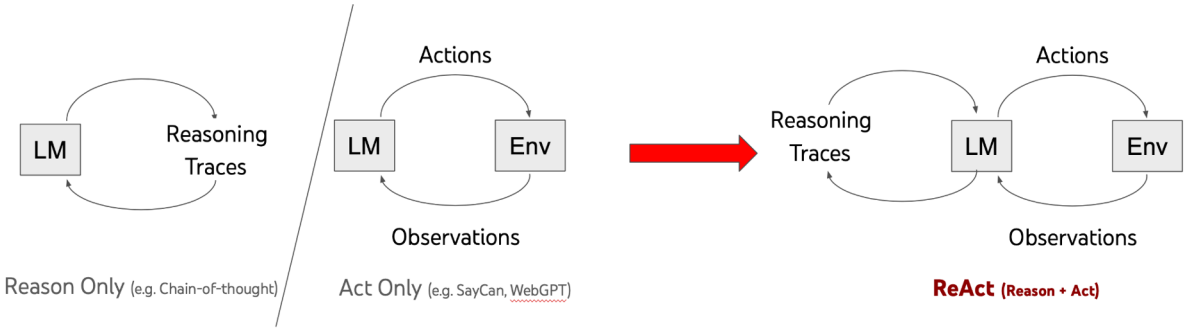
```
In [11]: prompt = f"""
Your task is to determine if the student's solution \
is correct or not.
To solve the problem do the following:
- First, work out your own solution to the problem including the final tot
- Then compare your solution to the student's solution \
and evaluate if the student's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

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

```
Let x be the size of the installation in square feet.

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

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

Is the student's solution the same as actual solution just
calculated:
No

Student grade:
Incorrect
```

There are many more prompting techniques.

- Chain-of-Thoughts (CoT)
- Self-Consistency
- ReAct





Retrieval Augmented Generation (RAG)

Hallucination issue

ChatGPT

พีรพล เวทีกูล คือใคร

พีรพล เวทีกูล เป็นนักข่าวที่มีชื่อเสียงในประเทศไทย โดยเฉพาะในวงการสื่อสารมวลชน เขาเป็นที่รู้จักในฐานะผู้ประกาศข่าวทางช่อง 3 และเป็นพิธีกรรายการที่มีความนิยมในหลายรายการโทรทัศน์ เช่น รายการข่าวพิเศษต่าง ๆ และรายการสารคดีต่าง ๆ ที่ออกอากาศในช่อง 3 ของประเทศไทย

ณรงค์เดช กีรติพรานนท์ คือใคร

ณรงค์เดช กีรติพรานนท์ เป็นนักการเมืองและนักธุรกิจชาวไทยที่มีชื่อเสียง โดยเป็นสมาชิกพรรคการเมืองชั้นนำในประเทศไทย นอกจากนี้เขายังเป็นผู้บริหารในธุรกิจต่าง ๆ

Retrieval Augmented Generation (RAG)


```
LawSections = f"""
มาตรา 1347 เจ้าของที่ดินอาจตัดรากไม้ซึ่งรุกเข้ามาจากที่ดินติดต่อและเอาไว้เสีย ถ้ากิ่งไม้ยื่นล้ำเข้ามา เมื่อเจ้าของที่ดินได้บอกผู้ครอบครองที่ดินติดต่อให้ตัดภายในเวลาอันสมควรแล้ว แต่ผู้นั้นไม่ตัด ท่านว่าเจ้าของที่ดินตัดเอาเสียก็ได้มาตรา 1348 ดอกผลแห่งต้นไม้ที่หล่นตามธรรมดาลงในที่ดินติดต่อ แปลงใด ท่านให้สันนิษฐานไว้ก่อนว่าเป็นดอกผลของที่ดินแปลงนั้น

มาตรา 1349 ที่ดินแปลงใดมีที่ดินแปลงอื่นล้อมอยู่จนไม่มีทางออกถึงทางสาธารณะได้ใช้ ท่านว่าเจ้าของที่ดินแปลงนั้นจะผ่านที่ดินซึ่งล้อมอยู่ไปสู่ทางสาธารณะได้

มาตรา 1350 ถ้าที่ดินแบ่งแยกหรือแบ่งโอนกันเป็นเหตุให้แปลงหนึ่งไม่ มีทางออกไปสู่ทางสาธารณะใช้ ท่านว่าเจ้าของที่ดินแปลงนั้นมีสิทธิเรียกร้อง เอาทางเดินตาม มาตรา ก่อนได้เฉพาะบนที่ดินแปลงที่ได้แบ่งแยกหรือแบ่งโอน กันและไม่ต้องเสียค่าทดแทน

มาตรา 1351 เจ้าของที่ดิน เมื่อบอกล่วงหน้าตามสมควรแล้ว อาจใช้ที่ดินติดต่อเพียงที่จำเป็นในการปลูกสร้างหรือซ่อมแซมรั้ว กำแพง หรือโรงเรือน ตรงหรือใกล้แนวเขตของตน แต่จะเข้าไปในเรือนที่อยู่ของเพื่อนบ้านข้างเคียงไม่ได้ เว้นแต่ได้รับความยินยอม
"""
```

Typical RAG Workflow

- **Preparation steps**
- Step 1: Split your documents into manageable-sized chunks.
- Step 2: Index your chunks into a database or knowledge base of some kind.

- **Inference steps**
- Step 3: Given a query, find the most similar documents in your knowledge base.
- Step 4: Feed the query and documents to the LLM to generate an answer.

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\n Hi 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'}}
```

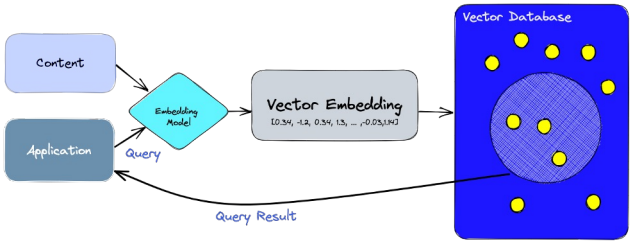
Name	Classes	Splits On	Adds Metadata	Description
Recursive	RecursiveCharacterTextSplitter, RecursiveJsonSplitter	A list of user defined characters		Recursively splits text. This splitting is trying to keep related pieces of text next to each other. This is the `recommended way` to start splitting text.
HTML	HTMLHeaderTextSplitter, HTMLSectionSplitter	HTML specific characters	✅	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	✅	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 Greg Kamradt
AI21 Semantic Text Splitter	AI21SemanticTextSplitter	✅	Identifies distinct topics that form coherent pieces of text and splits along those.	

https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/

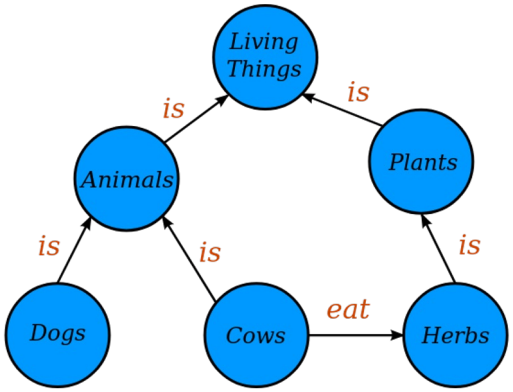
Typical RAG Workflow (2)

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

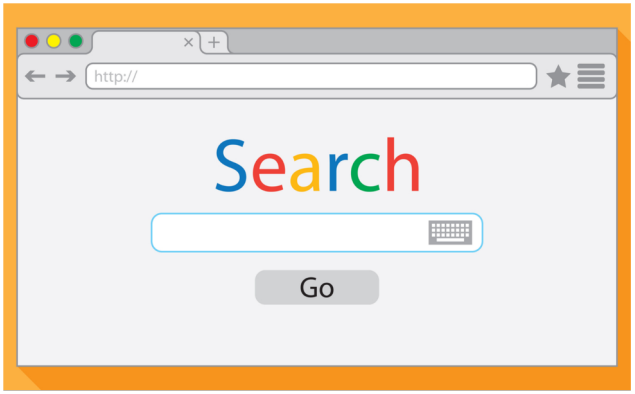
Vector Databases



Knowledge Graphs



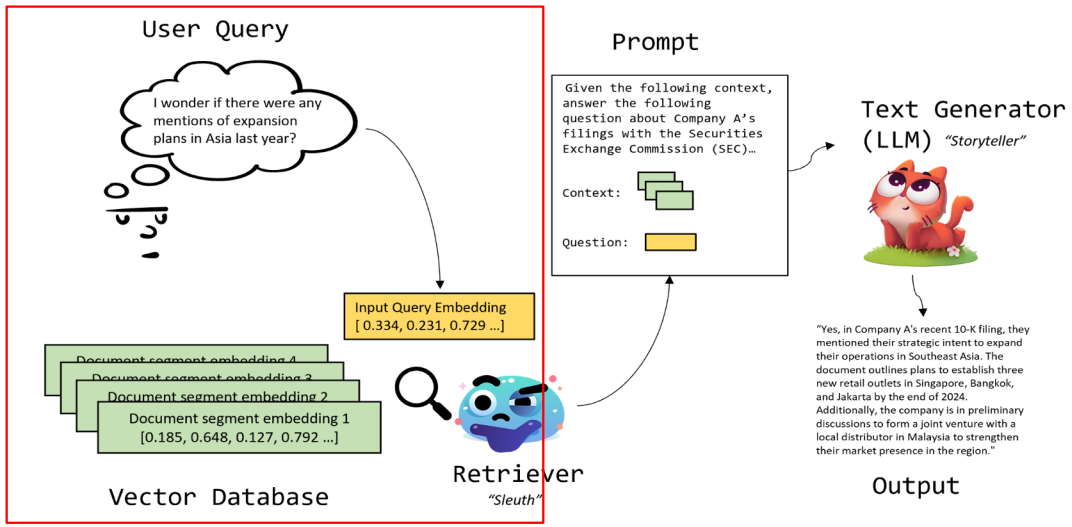
Search Engines



Typical RAG Workflow (3)

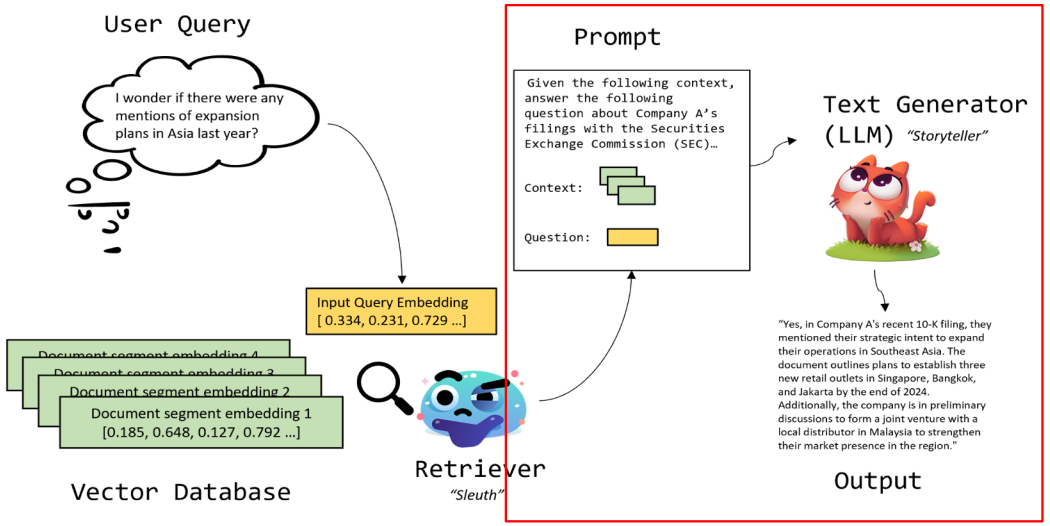
Step 3: Given a query, **find the most similar documents** in your knowledge base. E.g., perform vector search in a vector database.

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



Typical RAG Workflow (4)

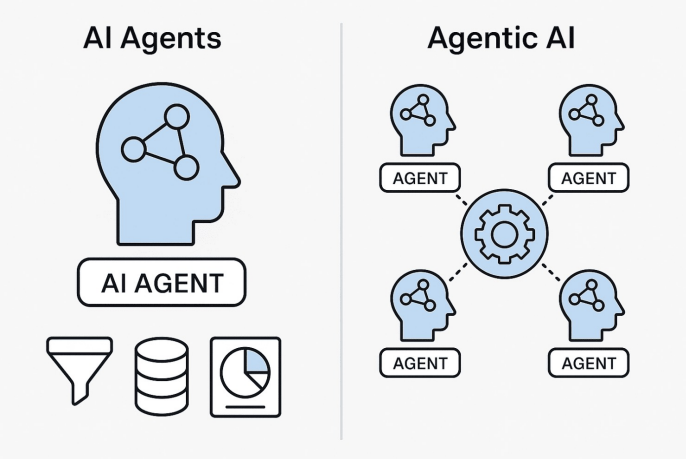
Step 4: Feed the query and documents to the LLM to generate an answer.

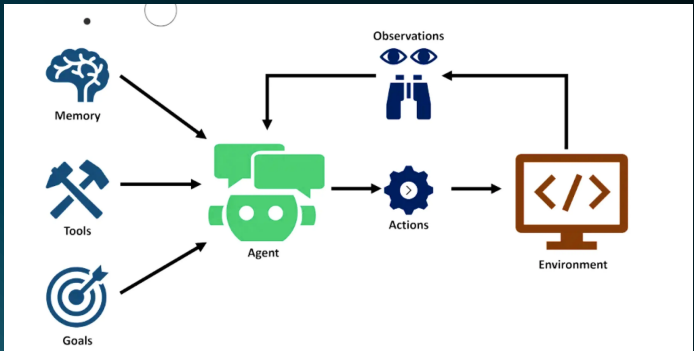


Agentic LLMs

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[*,1], Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]

[1]Department of Computer Science, Princeton University
[2]Google Research, Brain team
[1]{shunyuy,karthikn}@princeton.edu
[2]{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com

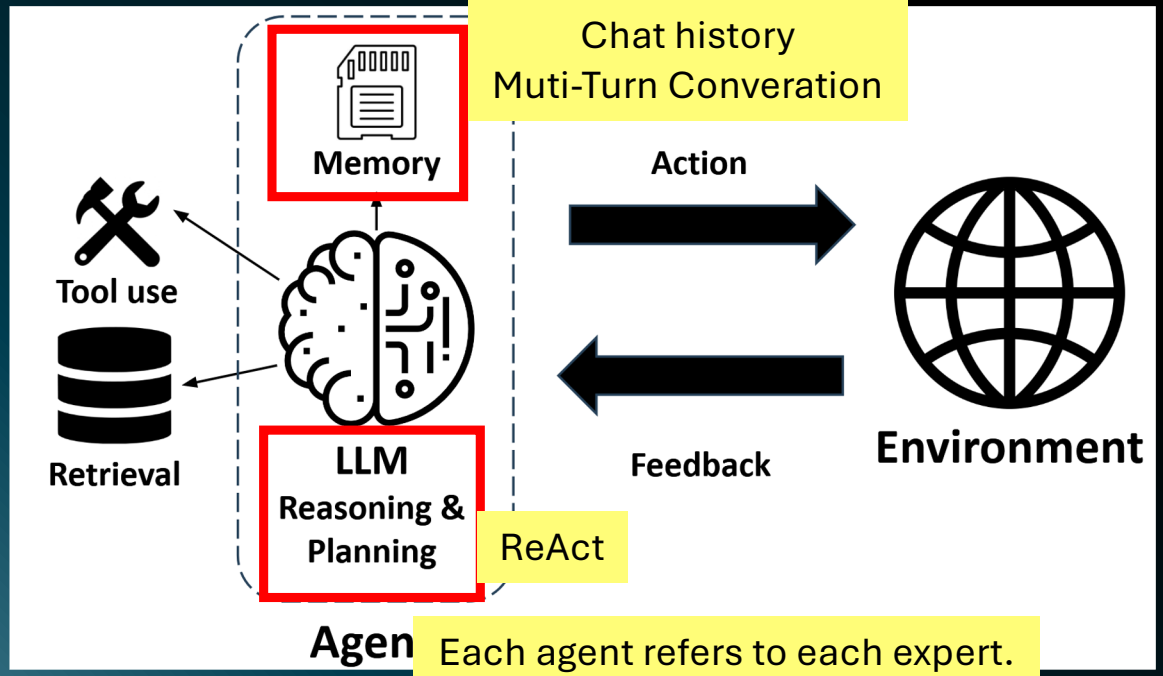
ABSTRACT

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

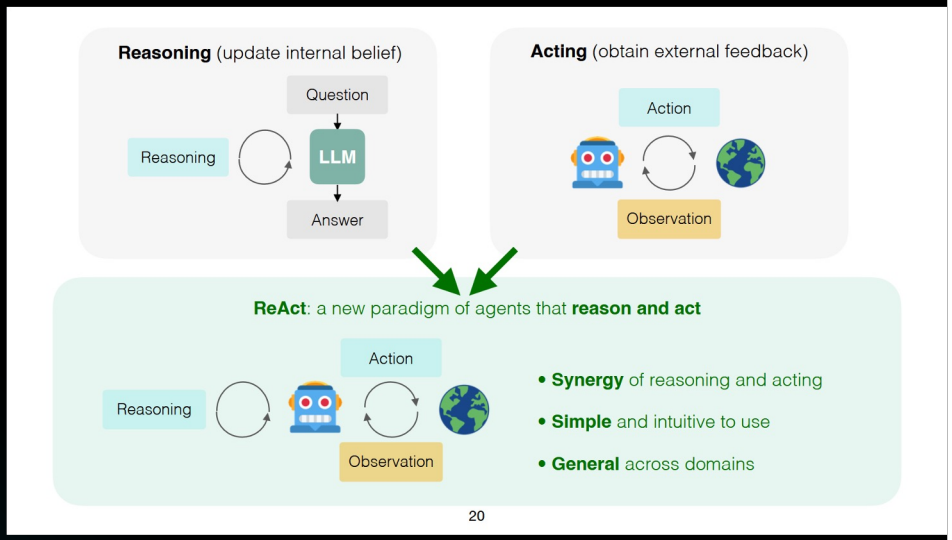
Agentic LLMs

ReAct = Reasoning (Thought) + Action + Observe

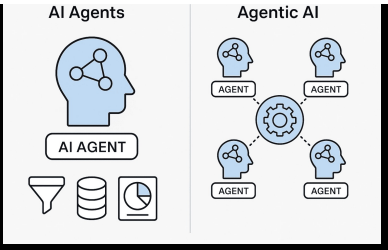
An agentic LLM is a large language model (LLM) that operates as an autonomous agent, capable of performing tasks, making decisions, and interacting with external tools or APIs to achieve the specified goals.



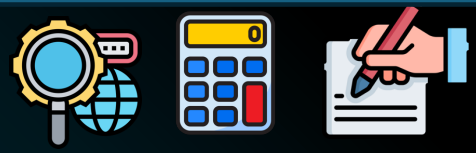
Reasoning: ReAct = Resoning + Action



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

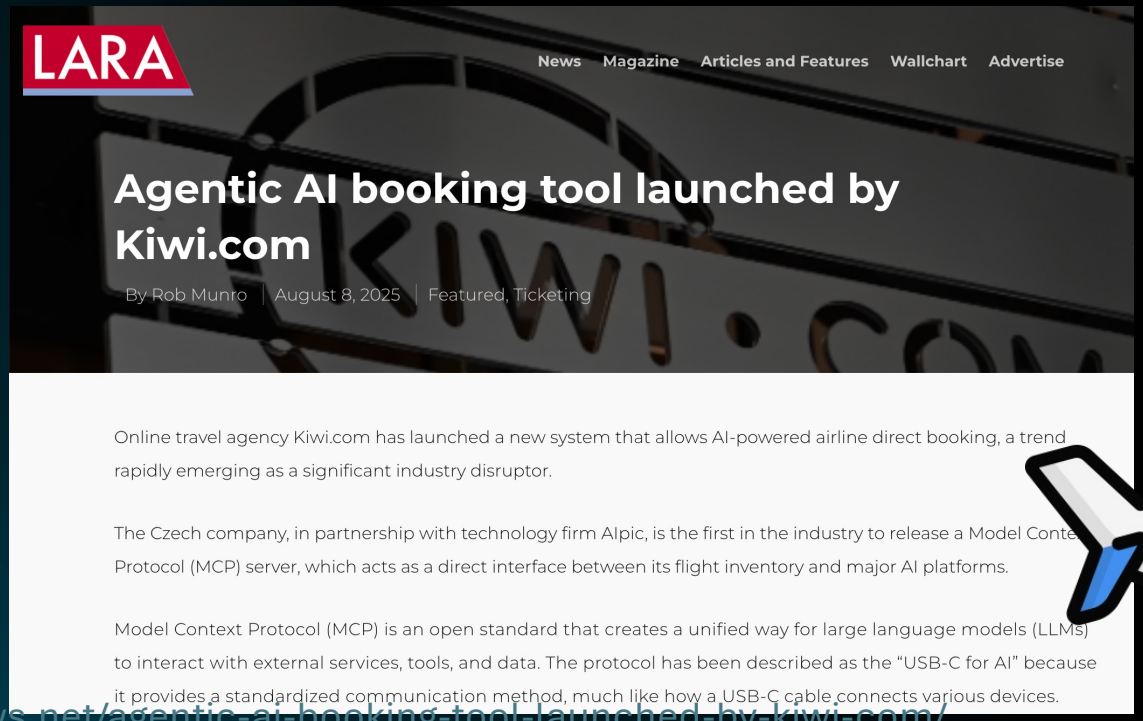
- **Step 3**
- Thought: Now I can calculate YoY growth: (1.45M – 1.20M) ÷ 1.20M
- Action: Call calc_yoy(current=1.45M, last=1.20M)
- Observation: YoY growth = 20.8%

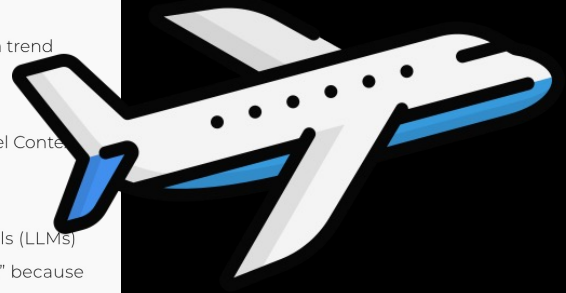
- **Step 4**
- Thought: I should summarize this result for the CFO.
- Action: Call summarize(revenue_now=1.45M, revenue_last=1.20M, yoy=20.8%)
- Observation: “Q2’25 revenue $1.45M vs Q2’24 $1.20M → YoY +20.8%”

- **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.



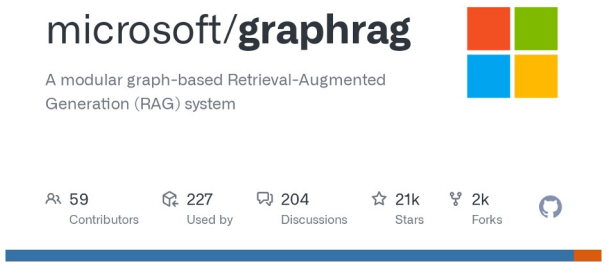


https://www.laranews.net/agentic-ai-booking-tool-launched-by-kiwi-com/

LLM Tools

LLM Tools

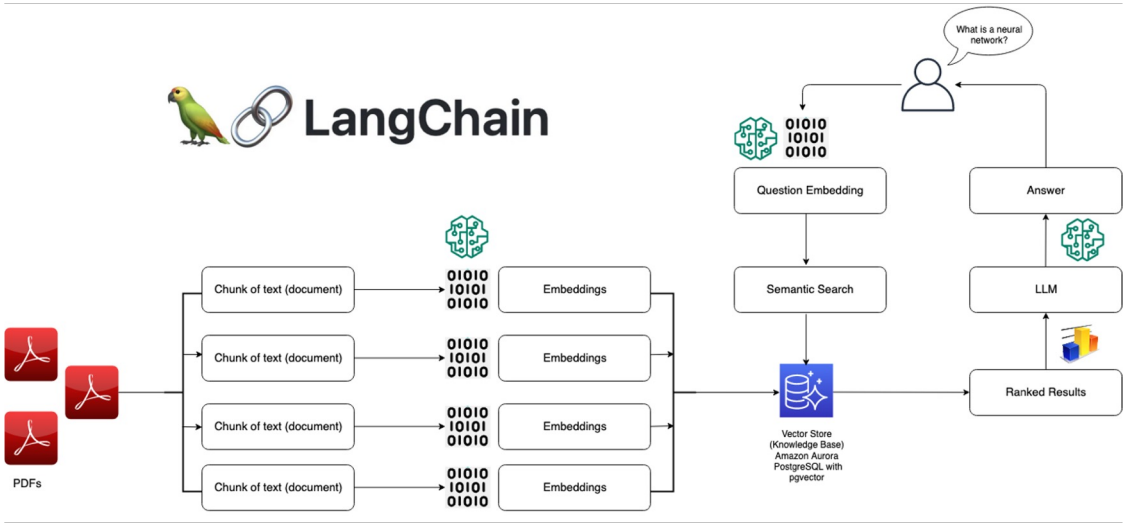
LLM Frameworks



Agentic Frameworks

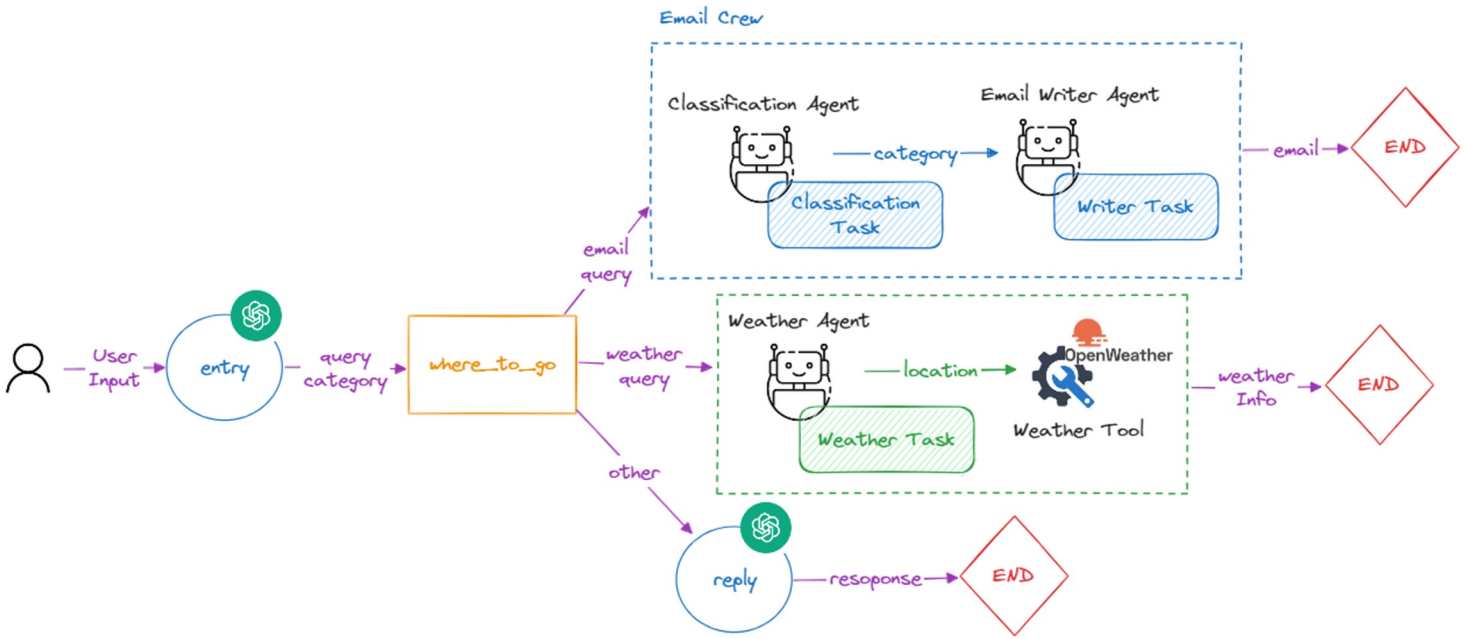
1) LangChain

LangChain is a framework that enables developers to build powerful LLM-based applications by managing external data connections, orchestrating multi-step interaction logic, and enhancing model capabilities such as context handling and tool execution, making it ideal for creating AI agents, chatbots, RAG systems, and complex automation workflows.



2) LangGraph

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



3) LangSmith (LLMOps)

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

1) Observability

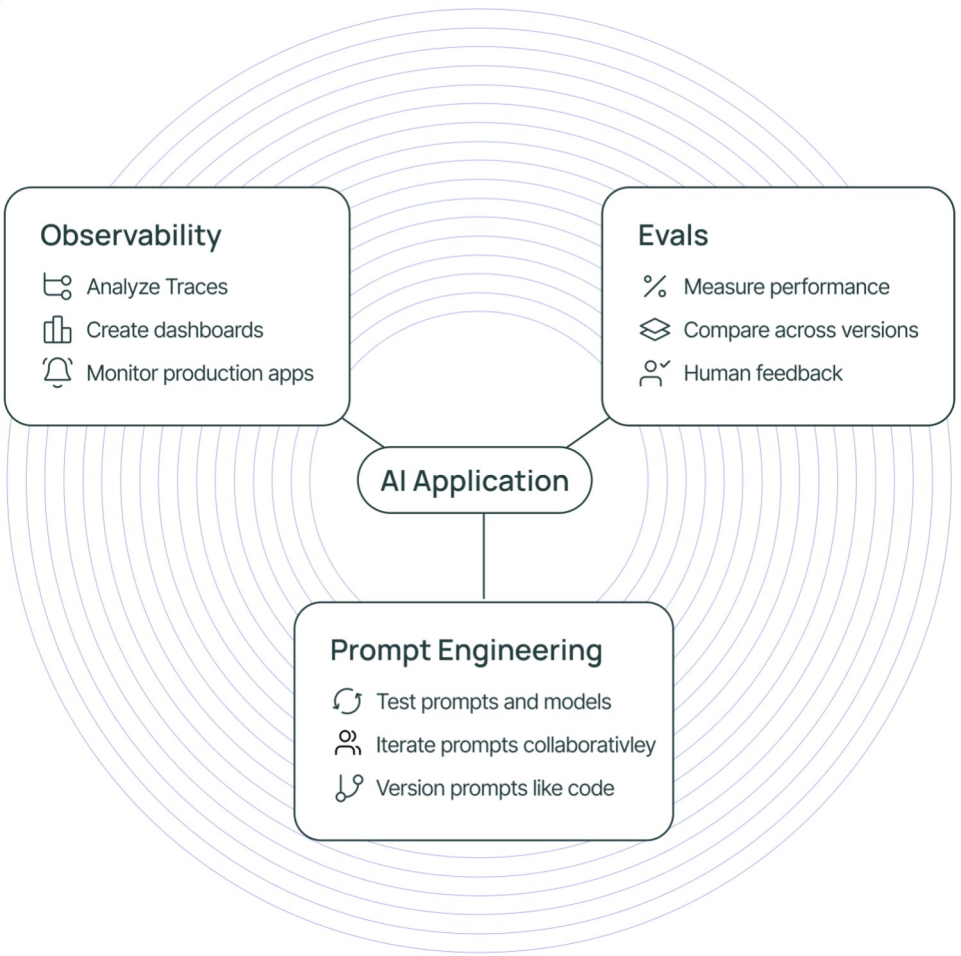
Analyze traces in LangSmith and configure metrics, dashboards, alerts based on these.

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.



https://www.langchain.com/langgraph

Please make sure to get all API keys before the lab.

1. gemini: https://aistudio.google.com/app/apikey
2. openai: https://platform.openai.com/account/api-keys
3. nvidia: https://build.nvidia.com/settings/api-keys

4. langsmith : smith.langchain.com
5. (Go to Developer (top right) → API Keys → Create API Key. Copy the key and save it securely.)

6. (optional) https://huggingface.co/settings/tokens (for 7_8 finetuning)

Q&A

Peerapon.v@chula.ac.th