

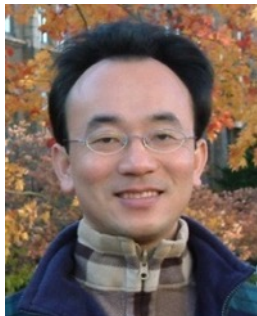
Python for Accounting Applications

Text Analytics with Generative AI and Python

1131PAA09

ACC2, NTPU (U2004) (Fall 2024)

Wed 6, 7, 8, (14:10-17:00) (9:10-12:00) (B3F10)



Min-Yuh Day, Ph.D,
Professor

Institute of Information Management, National Taipei University

<https://web.ntpu.edu.tw/~myday>



Syllabus

Week Date Subject/Topics

1 2024/09/11 Introduction to Python for Accounting Applications

2 2024/09/18 Python Programming and Data Science

3 2024/09/25 Foundations of Python Programming

4 2024/10/02 Data Structures

5 2024/10/09 Control Logic and Loops

6 2024/10/16 Functions and Modules; Files and Exception Handling

7 2024/10/23 Data Analytics and Visualization with Python

8 2024/10/30 ~~Midterm Project Report~~ (Self-Learning)

Syllabus

Week Date Subject/Topics

9 2024/11/06 Self-Learning

10 2024/11/13 Midterm Project Report

**11 2024/11/20 Obtaining Data From the Web with Python;
Statistical Analysis with Python**

12 2024/11/27 Machine Learning with Python

13 2024/12/04 Text Analytics with Generative AI and Python

14 2024/12/11 Applications of Accounting Data Analytics with Python

15 2024/12/18 Applications of ESG Data Analytics with Python

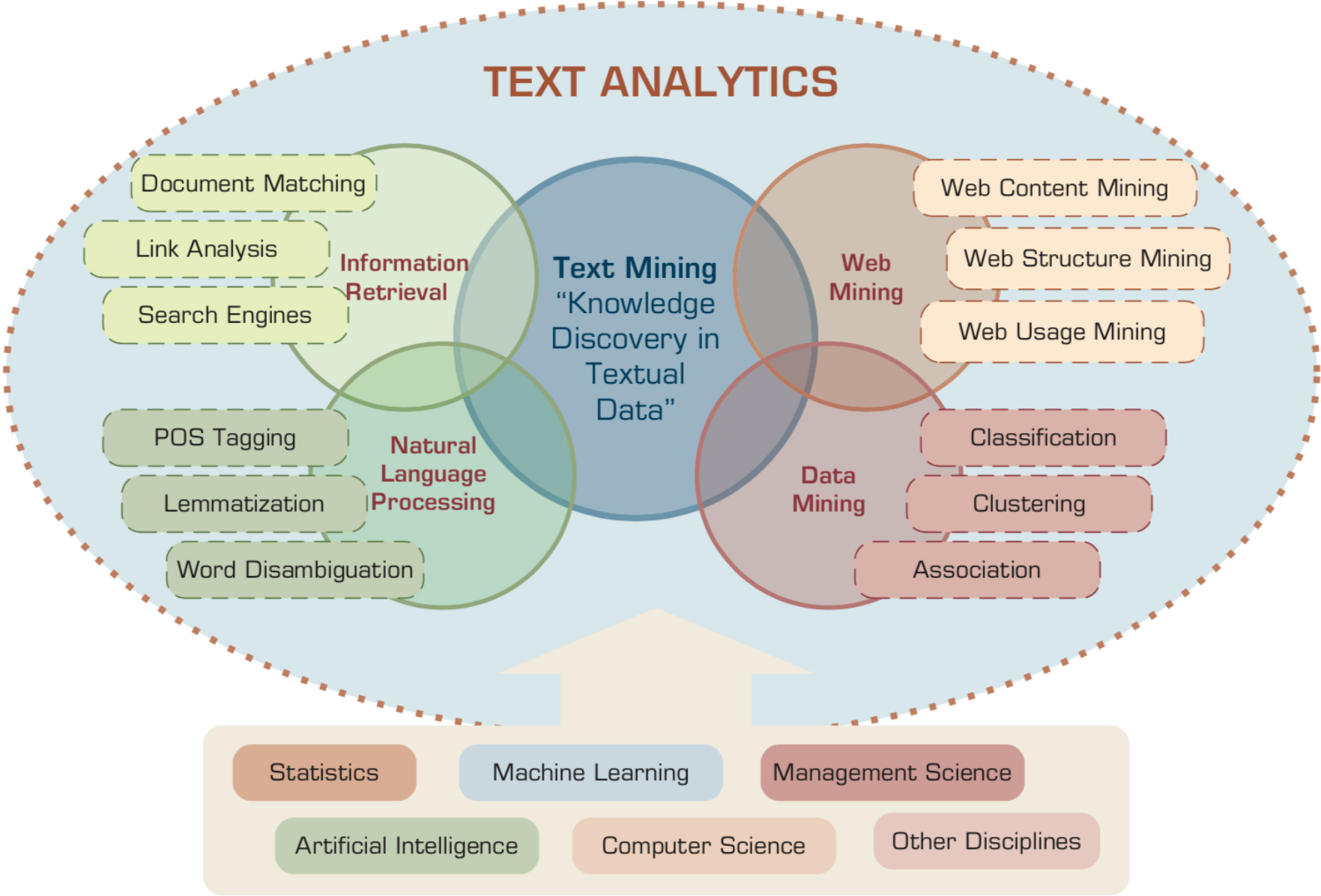
16 2024/12/25 Final Project Report

Text Analytics with Generative AI and Python

Outline

- **Text Analytics with Python**
- **Large Language Models (LLMs)**
- **Generative AI**

AI for Text Analytics



Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

O'REILLY®

Practical Natural Language Processing

A Comprehensive Guide to Building Real-World NLP Systems



Sowmya Vajjala,
Bodhisattwa Majumder,
Anuj Gupta & Harshit Surana

FOUNDATIONS

Covered in
Chapters 1 to 3



ML for NLP



NLP Pipelines



Data
Gathering



Multilingual
NLP



Text
Representation

CORE TASKS

Covered in
Chapters 3 to 7



Text
Classification



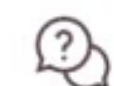
Information
Extraction



Conversational
Agents



Information
Retrieval



Question
Answering

GENERAL APPLICATIONS

Covered in
Chapters 4 to 7



Spam
Classification



Calendar Event
Extractor



Personal
Assistants



Search
Engines

JEOPARDY!

Jeopardy!

INDUSTRY SPECIFIC

Covered in
Chapters 8 to 10



Social Media
Analysis



Retail Data
Extraction



Health Records
Analysis



Financial
Analysis



Legal Entity
Extraction

AI PROJECT PLAYBOOK

Covered in
Chapters 2 & 11



Project
Processes



Best
Practices



Model
Iterations

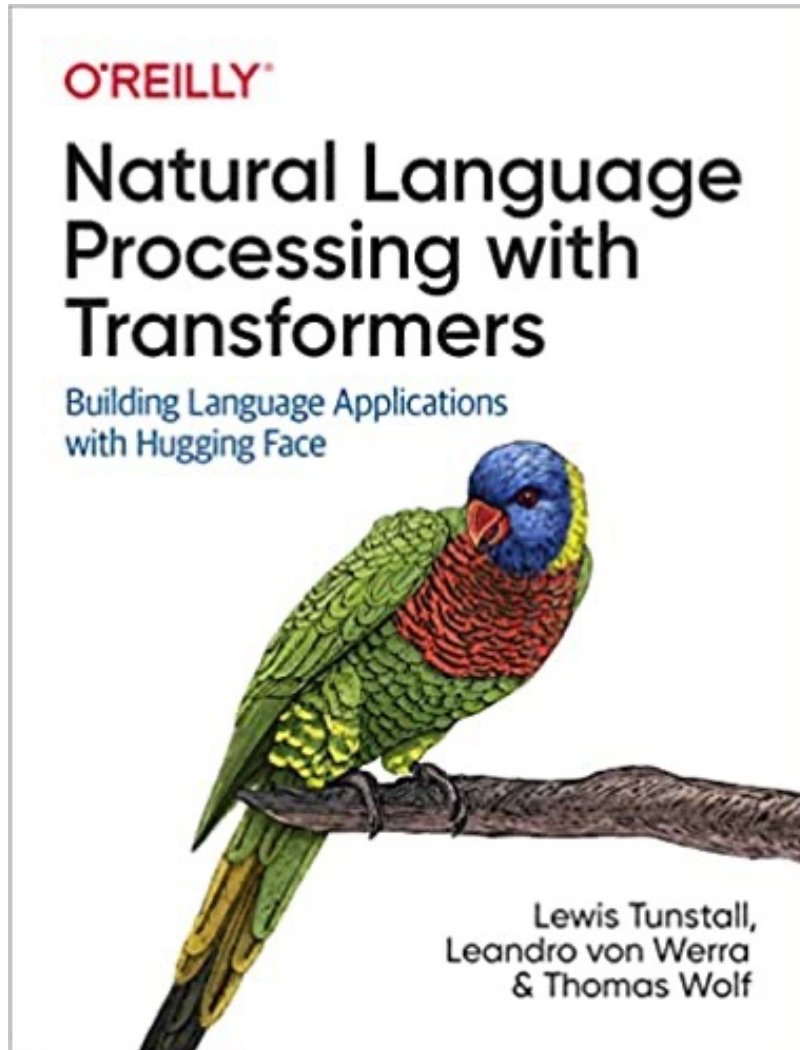


MLOps



AI Teams
& Hiring

NLP with Transformers Github Notebooks



Running on a cloud platform

To run these notebooks on a cloud platform, just click on one of the badges in the table below:

Chapter	Colab	Kaggle	Gradient	Studio Lab
Introduction	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Classification	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Transformer Anatomy	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Multilingual Named Entity Recognition	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Generation	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Summarization	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Question Answering	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Making Transformers Efficient in Production	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Dealing with Few to No Labels	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Training Transformers from Scratch	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Future Directions	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab

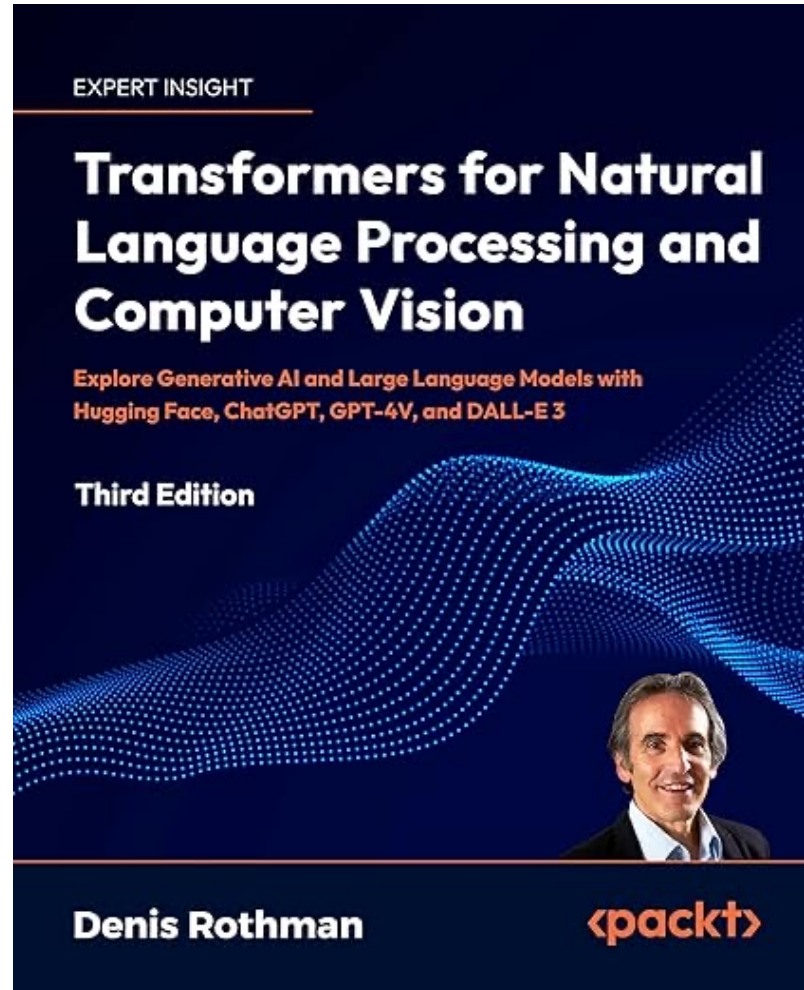
Nowadays, the GPUs on Colab tend to be K80s (which have limited memory), so we recommend using [Kaggle](#), [Gradient](#), or [SageMaker Studio Lab](#). These platforms tend to provide more performant GPUs like P100s, all for free!

<https://github.com/nlp-with-transformers/notebooks>

Denis Rothman (2024),

Transformers for Natural Language Processing and Computer Vision:

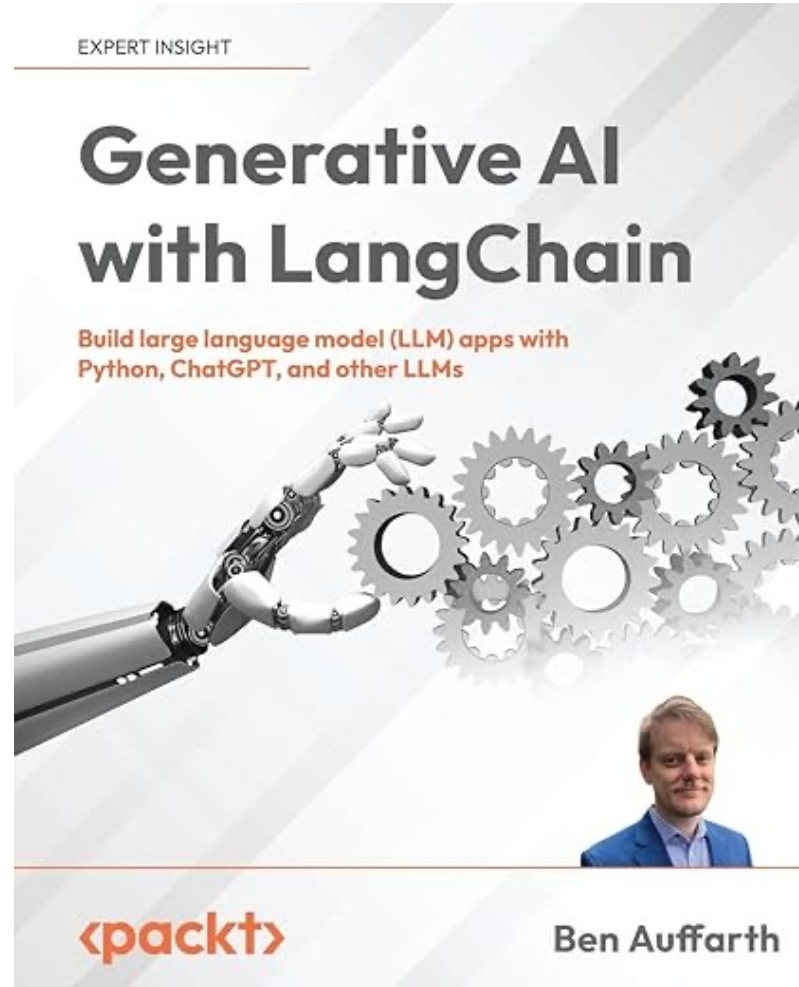
Explore Generative AI and Large Language Models with Hugging Face, ChatGPT, GPT-4V, and DALL-E 3,
3rd Edition, Packt Publishing



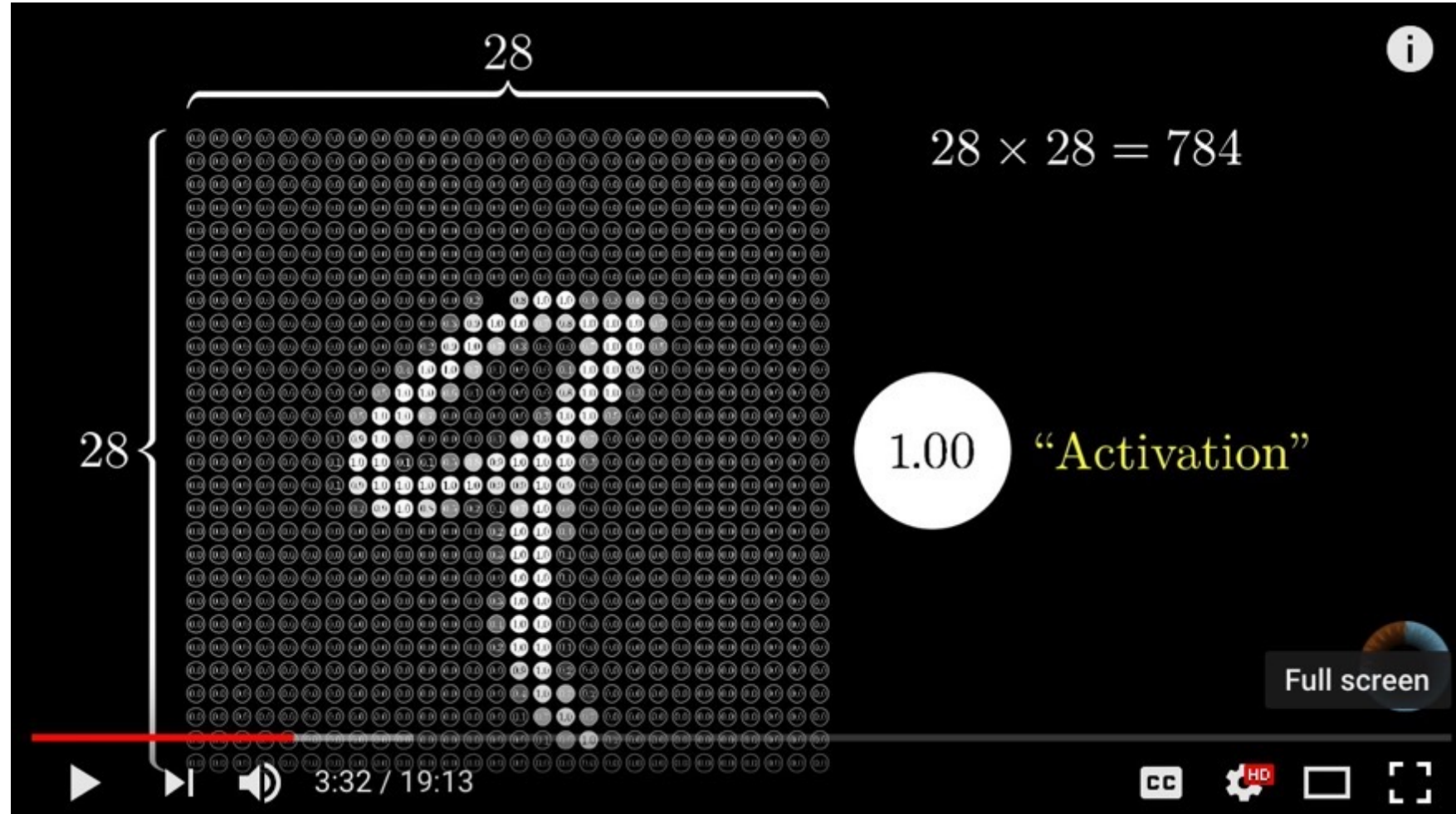
Ben Auffarth (2023),

Generative AI with LangChain:

Build large language model (LLM) apps with Python, ChatGPT and other LLMs,
Packt Publishing.



Neural Network and Deep Learning




Source: 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning,
<https://www.youtube.com/watch?v=aircAruvnKk>

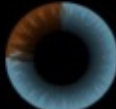
Gradient Descent

how neural networks learn

Average cost of all training data...

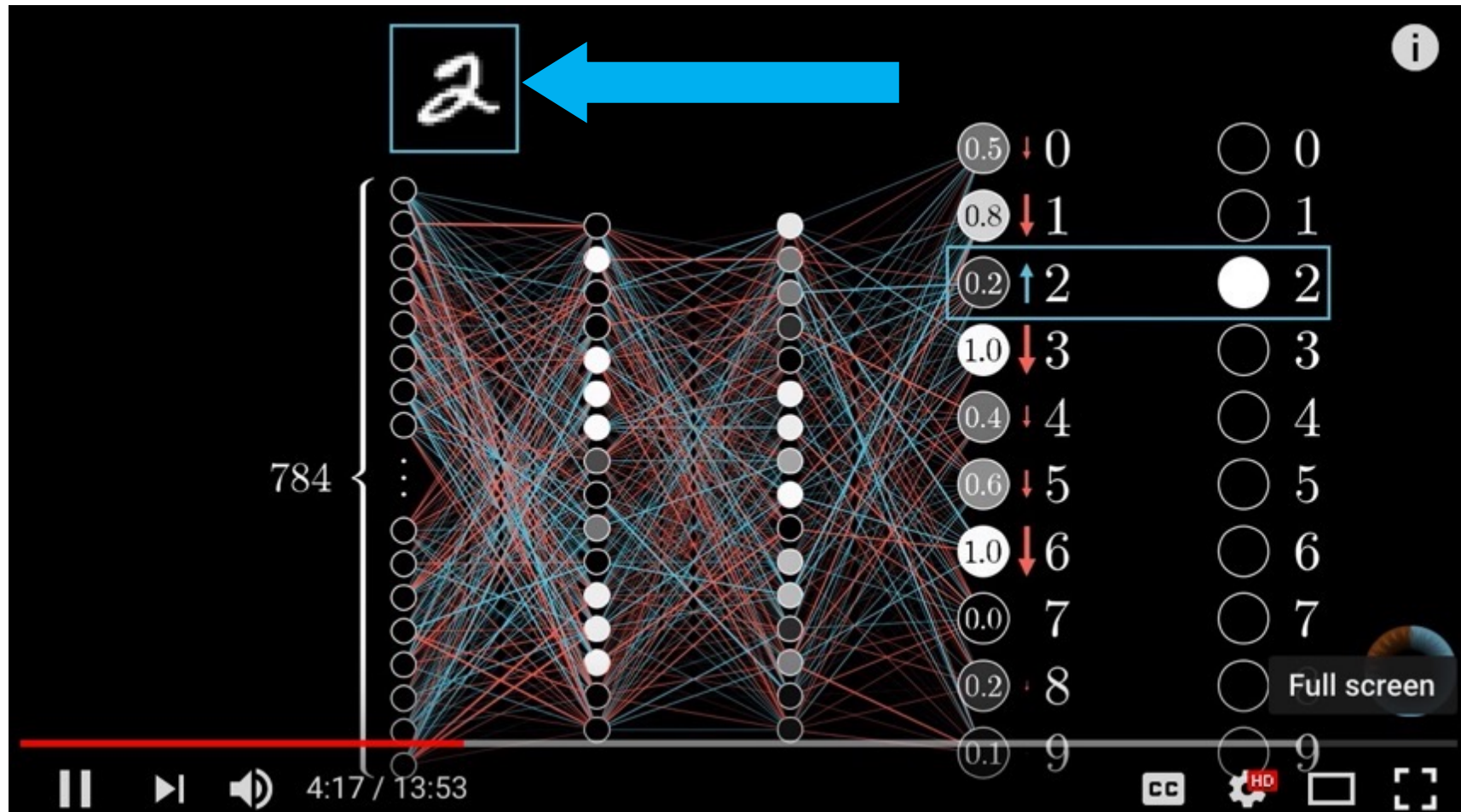
Cost of 

What's the "cost" of this difference?

Utter trash 

$(0.18 - 0.00)^2 +$	<input type="radio"/>	0
$(0.29 - 0.00)^2 +$	<input type="radio"/>	1
$(0.58 - 0.00)^2 +$	<input type="radio"/>	2
$(0.77 - 0.00)^2 +$	<input type="radio"/>	3
$(0.20 - 0.00)^2 +$	<input type="radio"/>	4
$(0.36 - 0.00)^2 +$	<input type="radio"/>	5
$(0.93 - 0.00)^2 +$	<input type="radio"/>	6
$(1.00 - 0.00)^2 +$	<input type="radio"/>	7
$(0.95 - 1.00)^2 +$	<input checked="" type="radio"/>	8
$(0.35 - 0.00)^2$	<input type="radio"/>	9

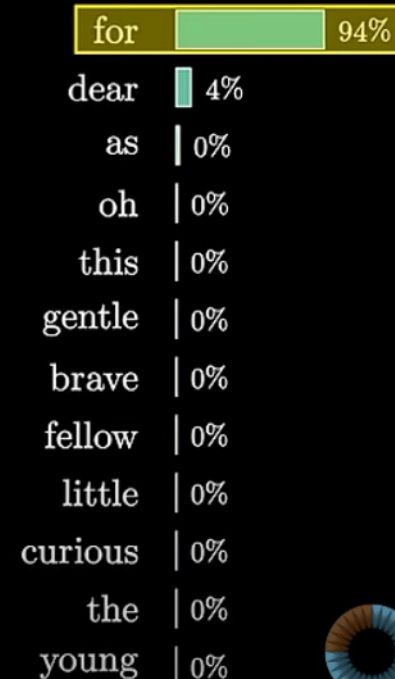
Backpropagation



Source: 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, <https://www.youtube.com/watch?v=llg3gGewQ5U>

Transformers (how LLMs work)

Behold, a wild pi creature, foraging in its native habitat of mathematical formulas and computer code! With its infinite digits and irrational tendencies, this strange creature is beloved by mathematicians and tech enthusiasts alike. Approach with caution, for attempting to calculate its exact value may lead to madness! But do not be afraid, **for**



Attention in Transformers

Query
1,572,864

$$\begin{bmatrix} -3.7 & +3.9 & -2.4 & -6.3 & -9.4 & -8.6 & +3.6 & -0.9 & \dots & +0.7 \\ +7.9 & +9.7 & -5.6 & +3.2 & -4.7 & -9.5 & +5.1 & -3.6 & \dots & -2.3 \\ +1.7 & +6.6 & +2.6 & +7.4 & -4.5 & +5.9 & -6.2 & +9.0 & \dots & +3.7 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -5.6 & +8.9 & +4.6 & -4.9 & -5.7 & +0.4 & -9.4 & -5.8 & \dots & -1.5 \end{bmatrix}$$

Key
1,572,864

$$\begin{bmatrix} -2.5 & -0.7 & -4.4 & +1.7 & +7.2 & -7.6 & +0.3 & -7.3 & \dots & +4.3 \\ -2.1 & +1.3 & -6.3 & -7.0 & -0.2 & -2.9 & +8.7 & +5.3 & \dots & +4.9 \\ +8.0 & -8.2 & +1.0 & +1.7 & +9.1 & -4.1 & -5.1 & -7.9 & \dots & -9.6 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ +8.5 & +3.4 & +5.6 & -4.3 & +1.7 & -8.6 & -0.3 & +9.5 & \dots & +7.5 \end{bmatrix}$$


Value
 $12,288 \times 12,288 = 150,994,944$

12,288

$$\begin{bmatrix} -3.2 & +9.1 & -5.3 & +8.9 & +8.7 & +5.9 & +2.6 & +7.4 & \dots & -4.1 \\ +6.9 & +2.3 & -9.6 & -3.0 & -7.0 & +9.5 & -0.4 & -0.1 & \dots & +2.8 \\ -2.6 & -7.2 & +6.4 & -6.1 & +0.2 & -5.5 & -8.0 & +7.2 & \dots & +9.4 \\ +9.1 & +8.0 & +5.4 & -3.3 & -8.3 & -1.8 & -5.3 & -7.3 & \dots & -8.8 \\ +4.5 & -9.7 & +5.4 & -7.0 & -8.3 & -8.1 & +3.4 & -5.0 & \dots & -1.6 \\ +1.1 & +7.1 & +4.5 & -4.5 & -7.3 & -8.8 & -3.9 & -4.7 & \dots & -0.9 \\ +3.6 & +3.9 & -4.3 & -2.4 & -6.3 & +5.7 & -8.8 & +3.9 & \dots & +5.5 \\ +5.5 & -4.8 & -2.5 & +1.7 & -4.5 & -2.6 & -6.0 & -0.8 & \dots & -9.0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ +5.9 & -8.4 & +0.4 & -3.8 & +1.5 & +9.1 & +2.9 & -9.2 & \dots & -1.4 \end{bmatrix} \begin{bmatrix} +0.2 \\ +0.7 \\ +3.6 \\ -4.4 \\ -7.3 \\ -2.1 \\ +9.0 \\ -6.2 \\ \vdots \\ +0.9 \end{bmatrix} = \begin{bmatrix} -198.6 \\ +73.1 \\ -28.2 \\ +119.4 \\ -4.4 \\ +215.7 \\ +91.8 \\ -29.1 \\ -5.6 \\ \vdots \\ -5.1 \end{bmatrix}$$

16:48 / 26:09 • Counting parameters >

How might LLMs store facts

 GPT-3

Total weights: i

175,181,291,520

Embedding	$12,288 \quad 50,257$ $d_embed * n_vocab$	$= 617,558,016$
Key	$128 \quad 12,288 \quad 96 \quad 96$ $d_query * d_embed * n_heads * n_layers$	$= 14,495,514,624$
Query	$128 \quad 12,288 \quad 96 \quad 96$ $d_query * d_embed * n_heads * n_layers$	$= 14,495,514,624$
Value	$128 \quad 12,288 \quad 96 \quad 96$ $d_value * d_embed * n_heads * n_layers$	$= 14,495,514,624$
Output	$12,288 \quad 128 \quad 96 \quad 96$ $d_embed * d_value * n_heads * n_layers$	$= 14,495,514,624$
Up-projection	$49,152 \quad 12,288 \quad 96$ $n_neurons * d_embed * n_layers$	$= 57,982,058,496$
Down-projection	$12,288 \quad 49,152 \quad 96$ $d_embed * n_neurons * n_layers$	$= 57,982,058,496$
Unembedding	$50,257 \quad 12,288$ $n_vocab * d_embed$	$= 617,558,016$

16:51 / 22:42 • Counting parameters >

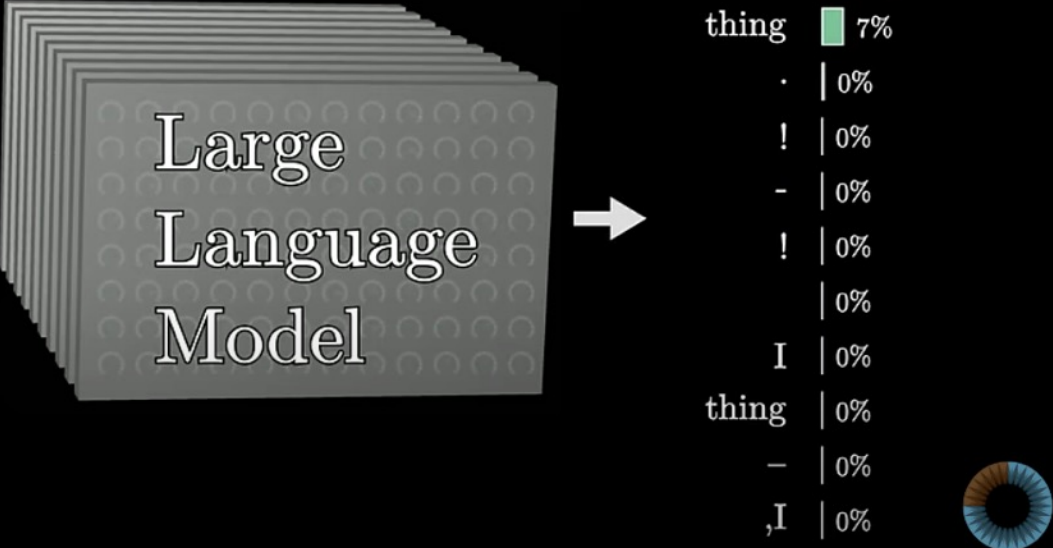
Large Language Models explained briefly

What follows is a conversation between a user and a helpful, very knowledgeable AI assistant.

User: Give me some ideas for what to do when visiting Santiago.

AI Assistant: Sure,

Large Language Model



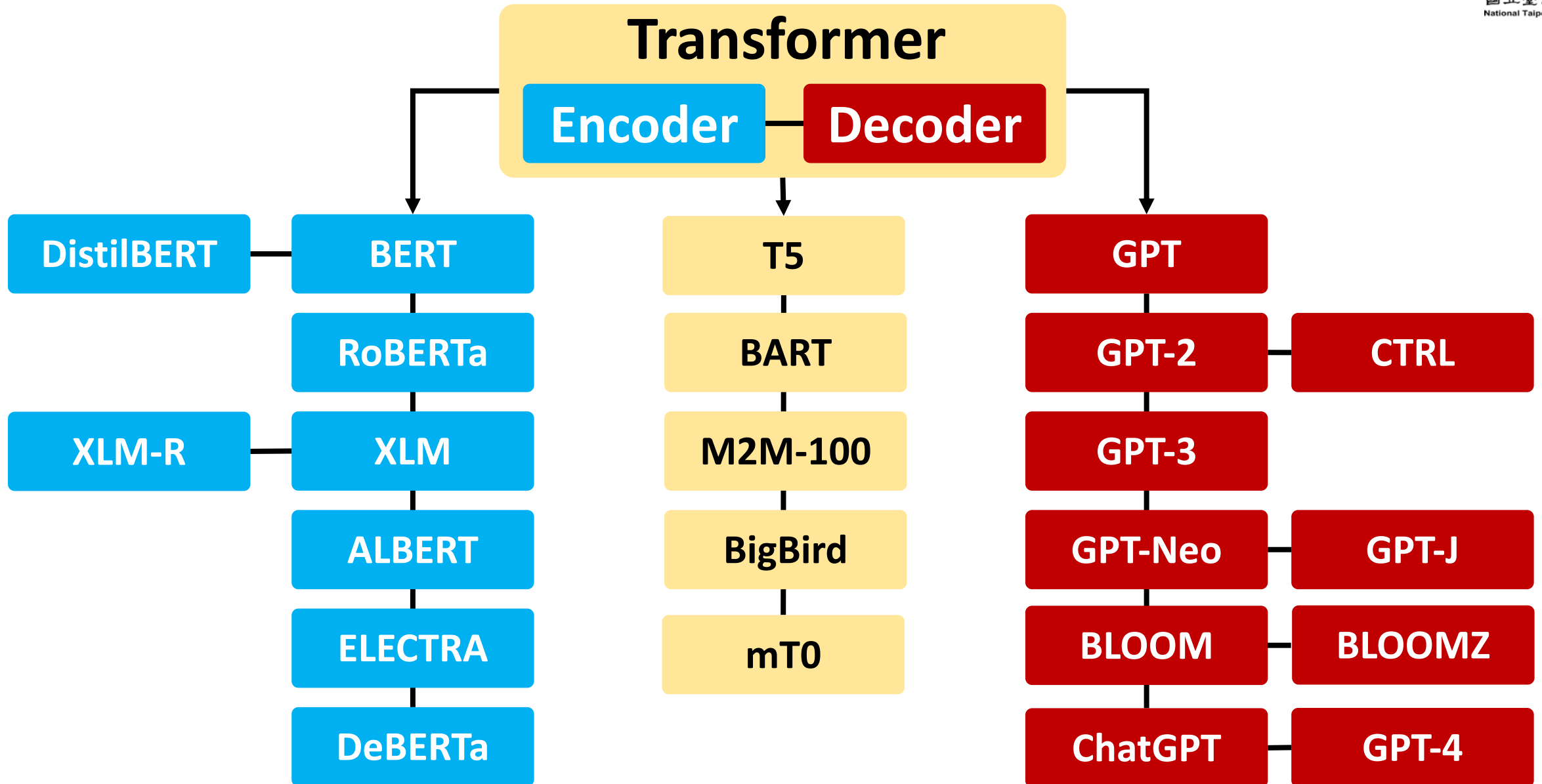
Token	Probability
,	53%
!	38%
thing	7%
.	0%
!	0%
-	0%
!	0%
.	0%
I	0%
thing	0%
-	0%
,I	0%

1:49 / 8:47 • What are large language models? >

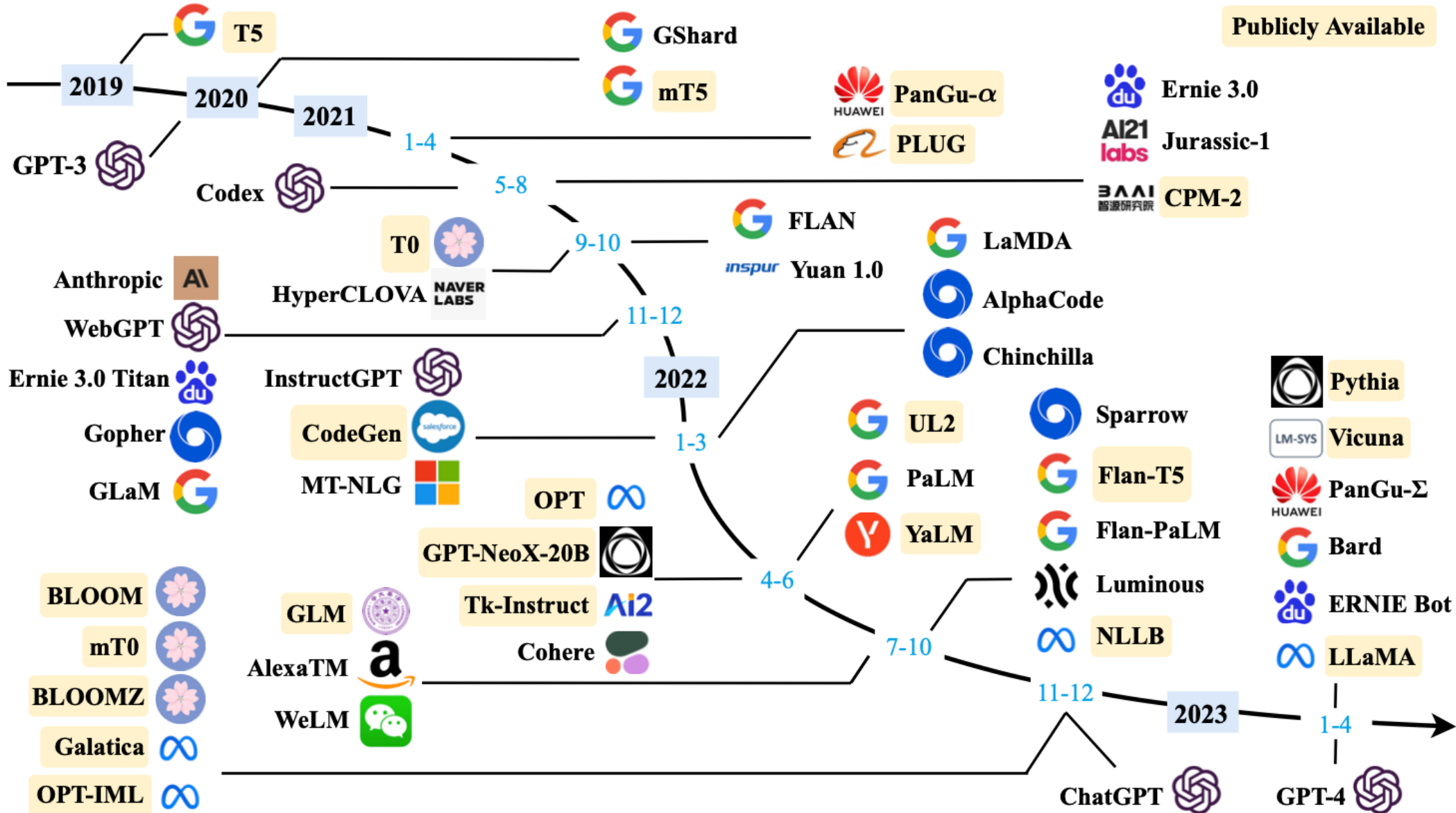
Source: 3Blue1Brown (2024), Large Language Models explained briefly, <https://www.youtube.com/watch?v=LPZh9BOjkQs>

Large Language Models (LLMs) Foundation Models

Transformer Models

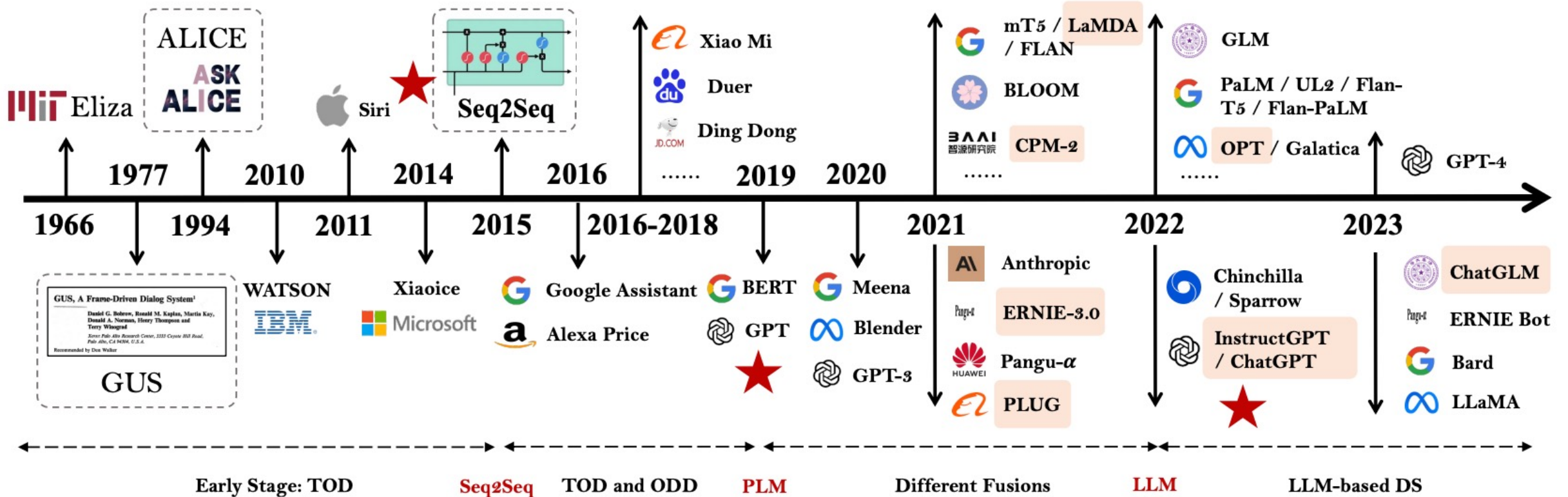


Large Language Models (LLMs) (larger than 10B)



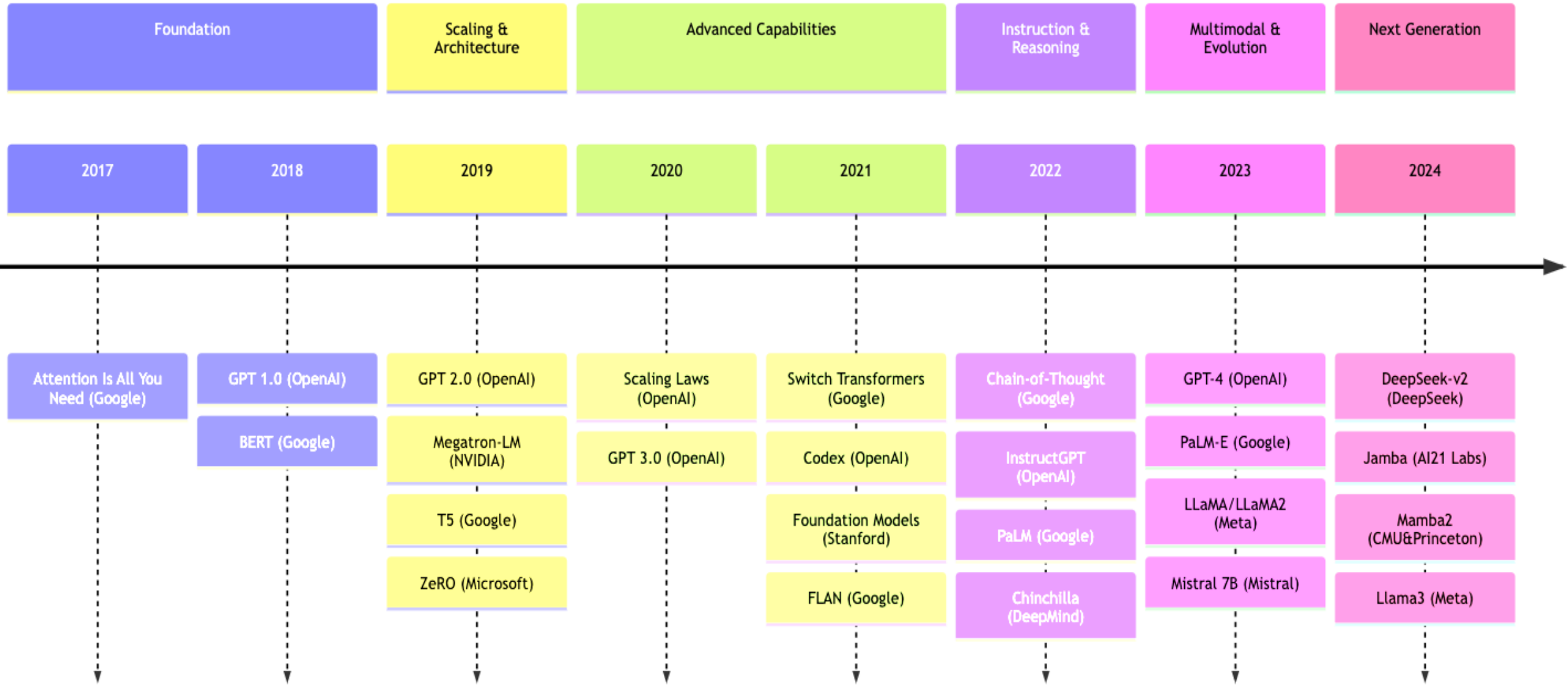
The Development of LM-based Dialogue Systems

- 1) Early Stage (1966 - 2015)
- 2) The Independent Development of TOD and ODD (2015 - 2019)
- 3) Fusions of Dialogue Systems (2019 - 2022)
- 4) LLM-based DS (2022 - Now)



Task-oriented DS (TOD), Open-domain DS (ODD)

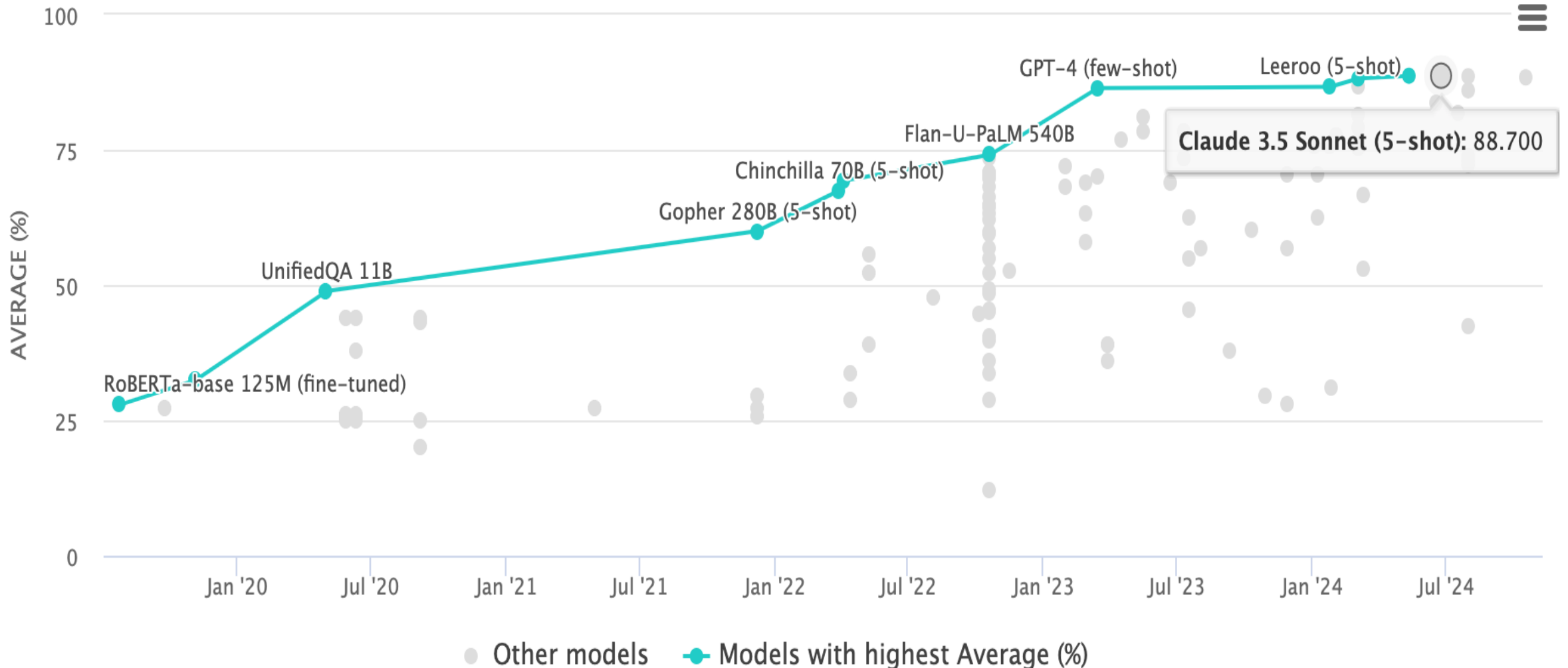
Major GenAI LLMs Research Milestones (2017-2024)



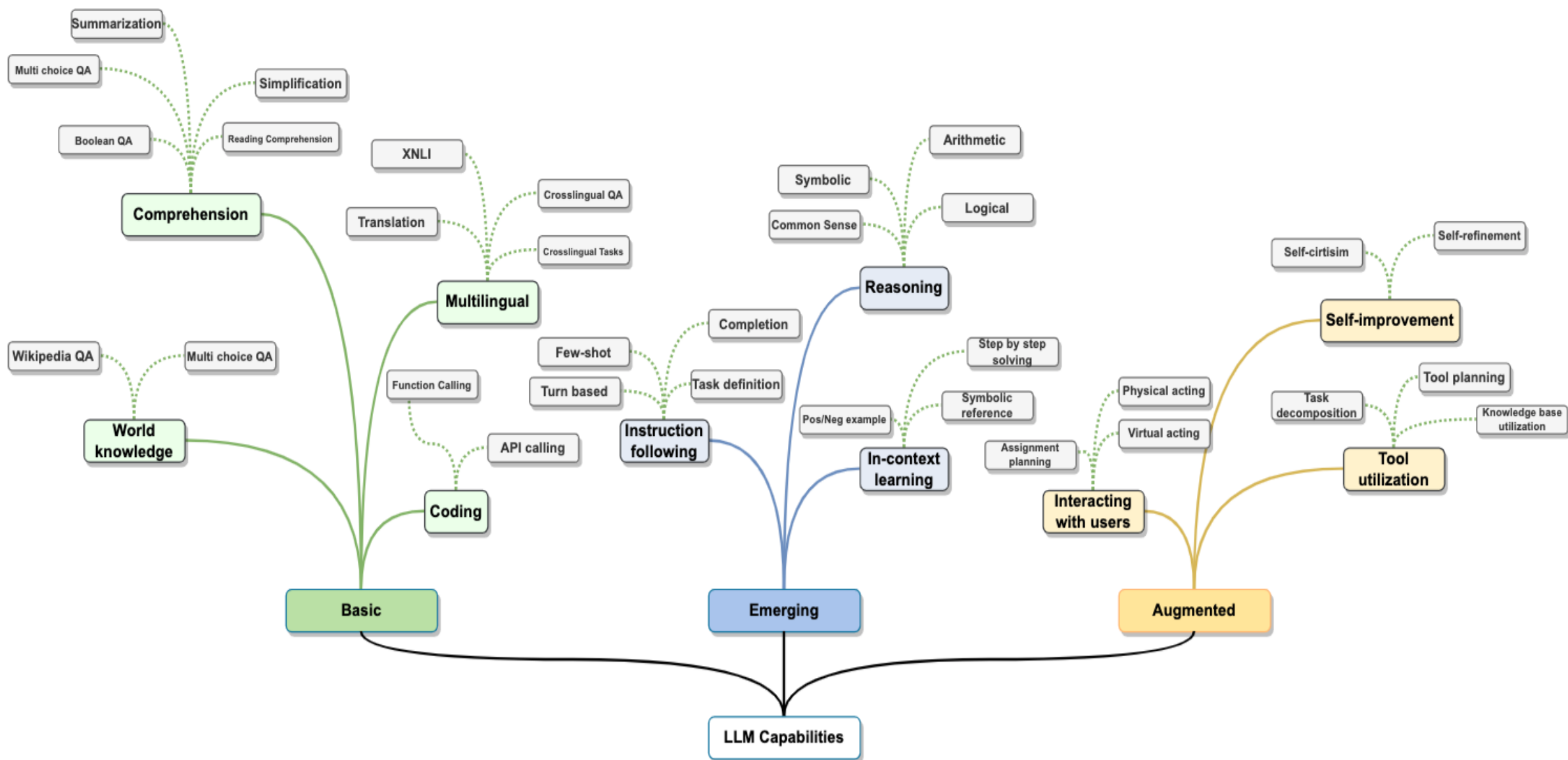
Multi-task Language Understanding on MMLU

GPT-4, Claude 3.5 Sonnet

Massive Multitask Language Understanding (MMLU)

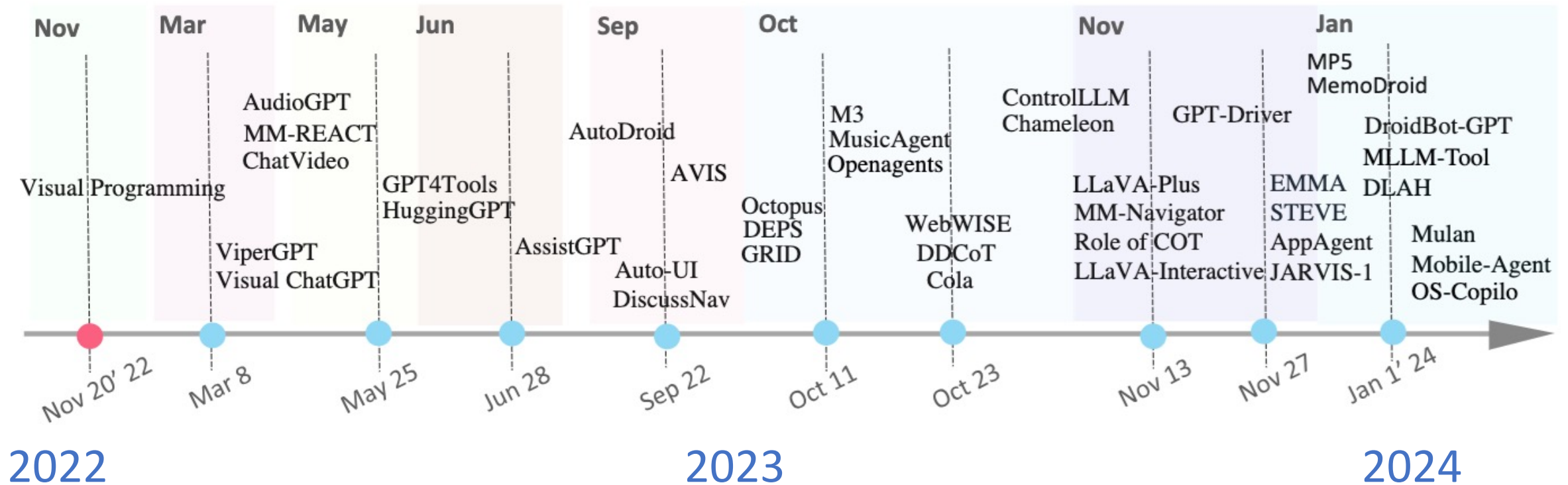


LLM Capabilities


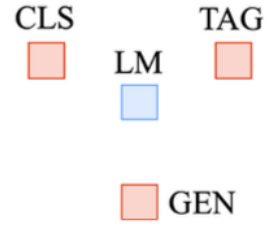
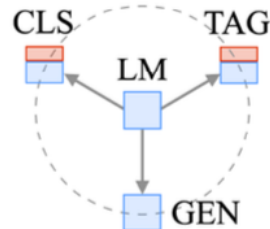
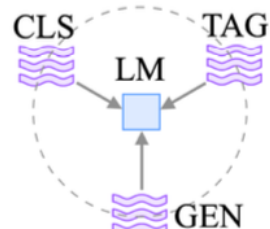


LLM-powered Multimodal Agents

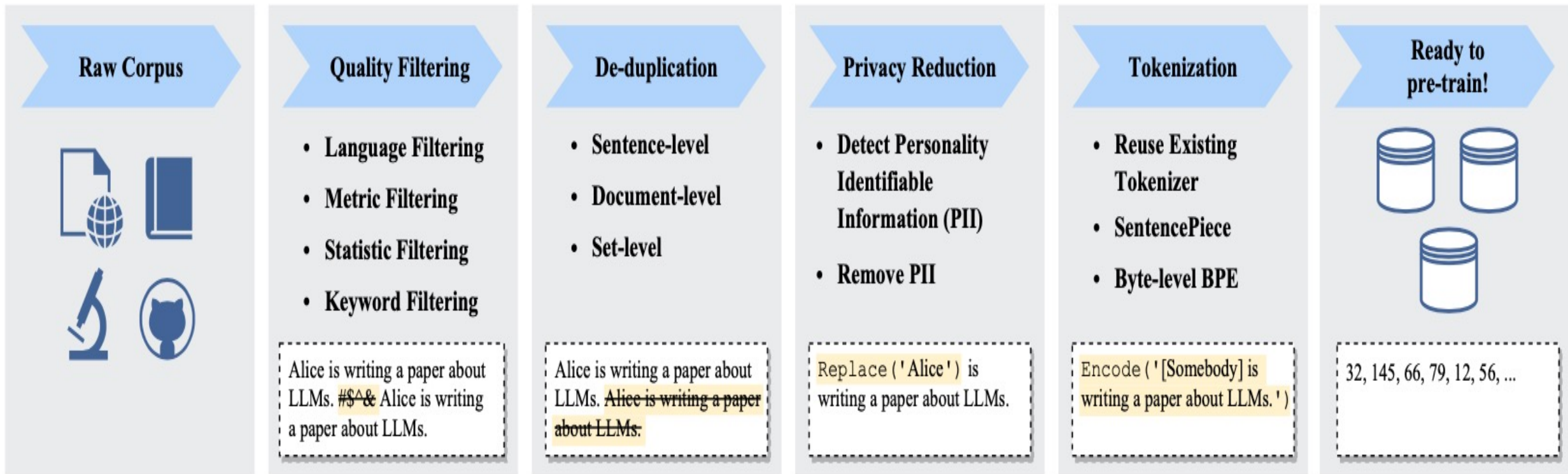
Large Multimodal Agents (LMAs)



Four Paradigms in NLP (LM)

Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Feature (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
Transfer Learning: Pre-training, Fine-Tuning (FT)		
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
GAI: Pre-train, Prompt, and Predict (Prompting)		
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

Typical Data Preprocessing Pipeline for Pre-training Large Language Models (LLMs)



Popular Generative AI

- **OpenAI ChatGPT (GPT-4o, GPT-4)**
- **Claude.ai (Claude 3.5)**
- **Google Gemini**
- **Meta Llama 3.2**
- **Mixtral Pixtral (mistral.ai)**
- **Chat.LMSys.org (lmarena.ai)**
- **Perplexity.ai**
- **Stable Diffusion**
- **Video: D-ID, Synthesia**
- **Audio: Speechify**

LMSYS Chatbot Arena Leaderboard

GPT-4o

Claude 3.5

Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization	License
1	2	Gemini-Exp-1121	1365	+8/-6	5625	Google	Proprietary
1	1	ChatGPT-4o-latest (2024-11-20)	1361	+4/-5	10658	OpenAI	Proprietary
3	5	Gemini-Exp-1114	1344	+4/-5	12778	Google	Proprietary
4	2	o1-preview	1334	+4/-4	27835	OpenAI	Proprietary
5	7	o1-mini	1308	+3/-4	31992	OpenAI	Proprietary
5	5	Gemini-1.5-Pro-002	1301	+5/-3	27336	Google	Proprietary
7	10	Grok-2-08-13	1289	+4/-3	52102	xAI	Proprietary
7	12	Yi-Lightning	1287	+4/-3	29336	01 AI	Proprietary
7	5	GPT-4o-2024-05-13	1285	+2/-2	111745	OpenAI	Proprietary
8	3	Claude 3.5 Sonnet (20241022)	1282	+4/-3	29454	Anthropic	Proprietary
10	17	Athene-v2-Chat-72B	1274	+8/-6	4354	NexusFlow	NexusFlow
11	18	GLM-4-Plus	1274	+5/-4	28133	Zhipu AI	Proprietary
11	19	GPT-4o-mini-2024-07-18	1273	+3/-3	51690	OpenAI	Proprietary
11	20	Gemini-1.5-Flash-002	1271	+4/-4	21071	Google	Proprietary
11	27	Llama-3.1-Nemotron-70B-Instruct	1269	+5/-6	7270	Nvidia	Llama 3.1
11	7	Claude 3.5 Sonnet (20240620)	1268	+2/-3	86632	Anthropic	Proprietary

Claude 3.5 Sonnet state-of-the-art vision

	Claude 3.5 Sonnet (new)	Claude 3.5 Haiku	Claude 3.5 Sonnet	GPT-4o*	GPT-4o mini*	Gemini 1.5 Pro	Gemini 1.5 Flash
Graduate level reasoning <i>GPQA (Diamond)</i>	65.0% 0-shot CoT	41.6% 0-shot CoT	59.4% 0-shot CoT	53.6% 0-shot CoT	40.2% 0-shot CoT	59.1% 0-shot CoT	51.0% 0-shot CoT
Undergraduate level knowledge <i>MMLU Pro</i>	78.0% 0-shot CoT	65.0% 0-shot CoT	75.1% 0-shot CoT	—	—	75.8% 0-shot CoT	67.3% 0-shot CoT
Code <i>HumanEval</i>	93.7% 0-shot	88.1% 0-shot	92.0% 0-shot	90.2% 0-shot	87.2% 0-shot	—	—
Math problem-solving <i>MATH</i>	78.3% 0-shot CoT	69.2% 0-shot CoT	71.1% 0-shot CoT	76.6% 0-shot CoT	70.2% 0-shot CoT	86.5% 4-shot CoT	77.9% 4-shot CoT
High school math competition <i>AIME 2024</i>	16.0% 0-shot CoT	5.3% 0-shot CoT	9.6% 0-shot CoT	9.3% 0-shot CoT	—	—	—
Visual Q/A <i>MMMU</i>	70.4% 0-shot CoT	—	68.3% 0-shot CoT	69.1% 0-shot CoT	59.4% 0-shot CoT	65.9% 0-shot CoT	62.3% 0-shot CoT
Agentic coding <i>SWE-bench Verified</i>	49.0%	40.6%	33.4%	—	—	—	—
Agentic tool use <i>TAU-bench</i>	Retail 69.2% Airline 46.0%	Retail 51.0% Airline 22.8%	Retail 62.6% Airline 36.0%	—	—	—	—

* Our evaluation tables exclude OpenAI's o1 model family as they depend on extensive pre-response computation time, unlike typical models. This fundamental difference makes performance comparisons difficult.

Llama 3.2 90B vision LLMs

Modality	Category Benchmark	Llama 3.2 11B	Llama 3.2 90B	Claude 3 – Haiku	GPT-4o-mini
Image	College-level Problems and Mathematical Reasoning MMMU (val, 0-shot CoT, micro avg accuracy)	50.7	60.3	50.2	59.4
	MMMU-Pro, Standard (10 opts, test)	33.0	45.2	27.3	42.3
	MMMU-Pro, Vision (test)	23.7	33.8	20.1	36.5
	MathVista (testmini)	51.5	57.3	46.4	56.7
	Charts and Diagram Understanding ChartQA (test, 0-shot CoT relaxed accuracy)*	83.4	85.5	81.7	—
	AI2 Diagram (test)*	91.1	92.3	86.7	—
	DocVQA (test, ANLS)*	88.4	90.1	88.8	—
	General Visual Question Answering VQAv2 (test)	75.2	78.1	—	—
Text	General MMLU (0-shot, CoT)	73.0	86.0	75.2 (5-shot)	82.0
	Math MATH (0-shot, CoT)	51.9	68.0	38.9	70.2
	Reasoning GPQA (0-shot, CoT)	32.8	46.7	33.3	40.2
	Multilingual MGSM (0-shot, CoT)	68.9	86.9	75.1	87.0

Mistral Pixtral Large (124B)

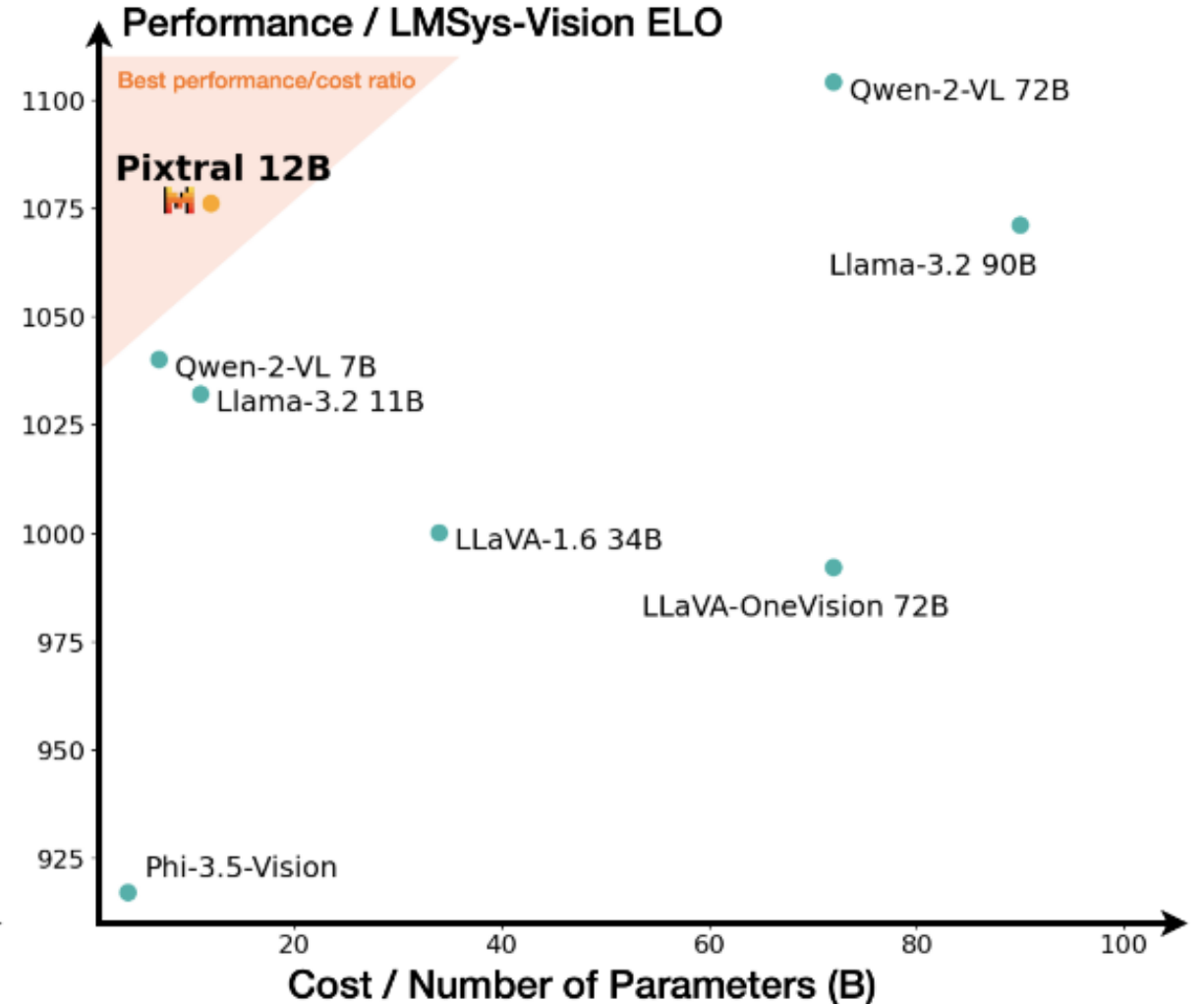
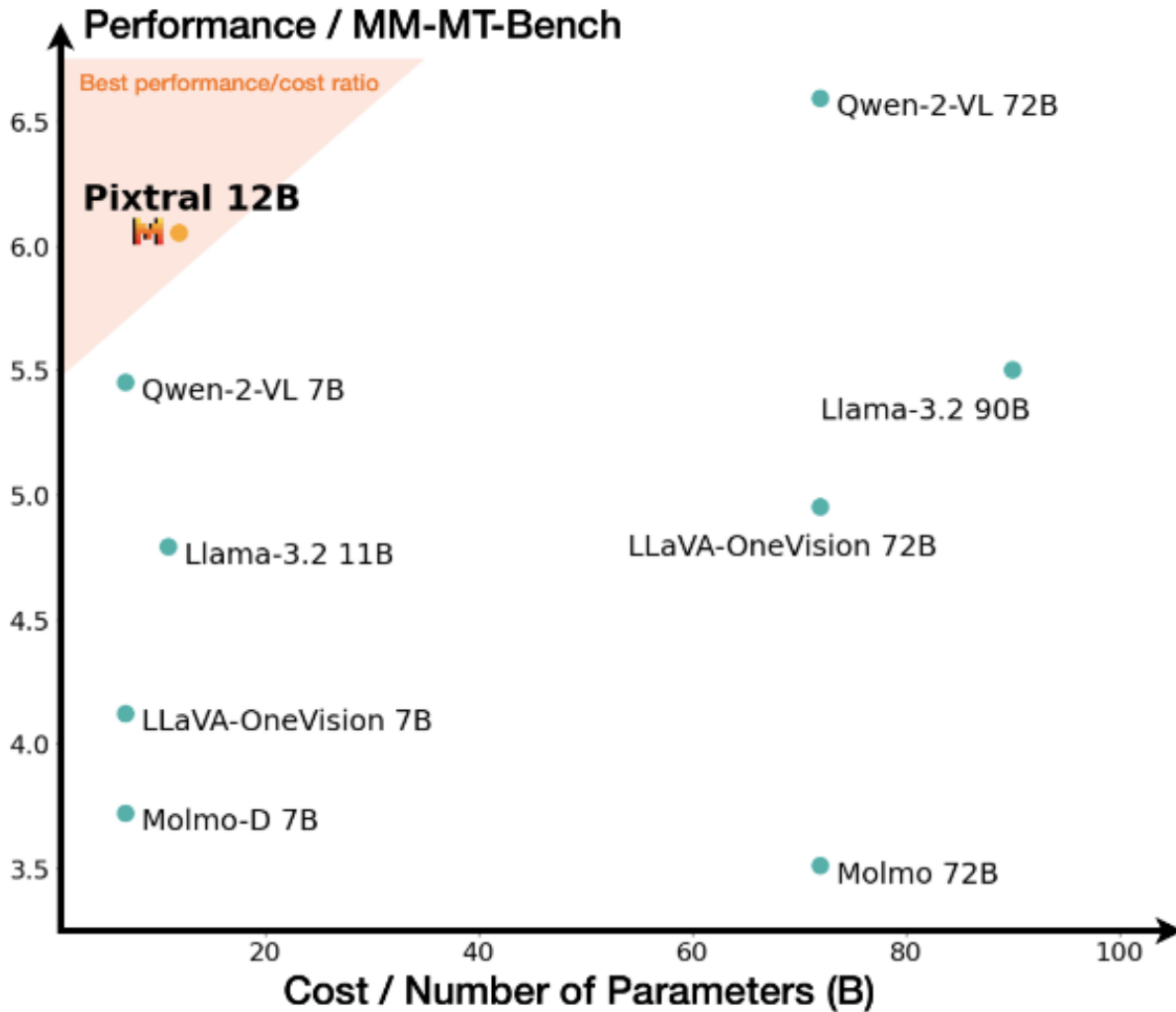
Frontier-class multimodal performance

Model	Mathvista (CoT)	MMMU (CoT)	ChartQA (CoT)	DocVQA (ANLS)	VQAv2 (VQA Match)	AI2D (BBox)	MM MT-Bench	
Open Weights	Pixtral Large (124B)	69.4	64.0	88.1	93.3	80.9	93.8	7.4
	Llama-3.2 90B (measured)	49.1	53.7	70.8	85.7	67.0	-	5.5
	Llama-3.2 90B (reported)	57.3	60.3	85.8	90.1	80.2	92.3	-
Closed	Gemini-1.5 Pro (measured)	67.8	66.3	83.8	92.3	70.6	94.6	6.8
	Gemini-1.5 Pro (reported)	68.1	65.9	-	-	-	-	-
	GPT-4o (measured)	65.4	68.6	85.2	88.5	76.4	93.2	6.7
	GPT-4o (reported)	63.8	69.1	85.7	92.8	-	-	-
	Claude-3.5 Sonnet (measured)	67.1	68.4	89.1	88.6	69.5	76.9	7.3
	Claude-3.5 Sonnet (reported)	70.7	70.4	90.8	94.2	-	95.3	-
Unreleased	Llama-3.1 505B (reported)	-	64.5	85.8	92.6	80.2	94.1	-
	Grok-2 (reported)	69.0	66.1	-	93.6	-	-	-

Specific model versions evaluated: Claude-3.5 Sonnet (new) [Oct 24], Gemini-1.5 Pro (002) [Sep 24], GPT-4o (2024-08-06) [Aug 24]

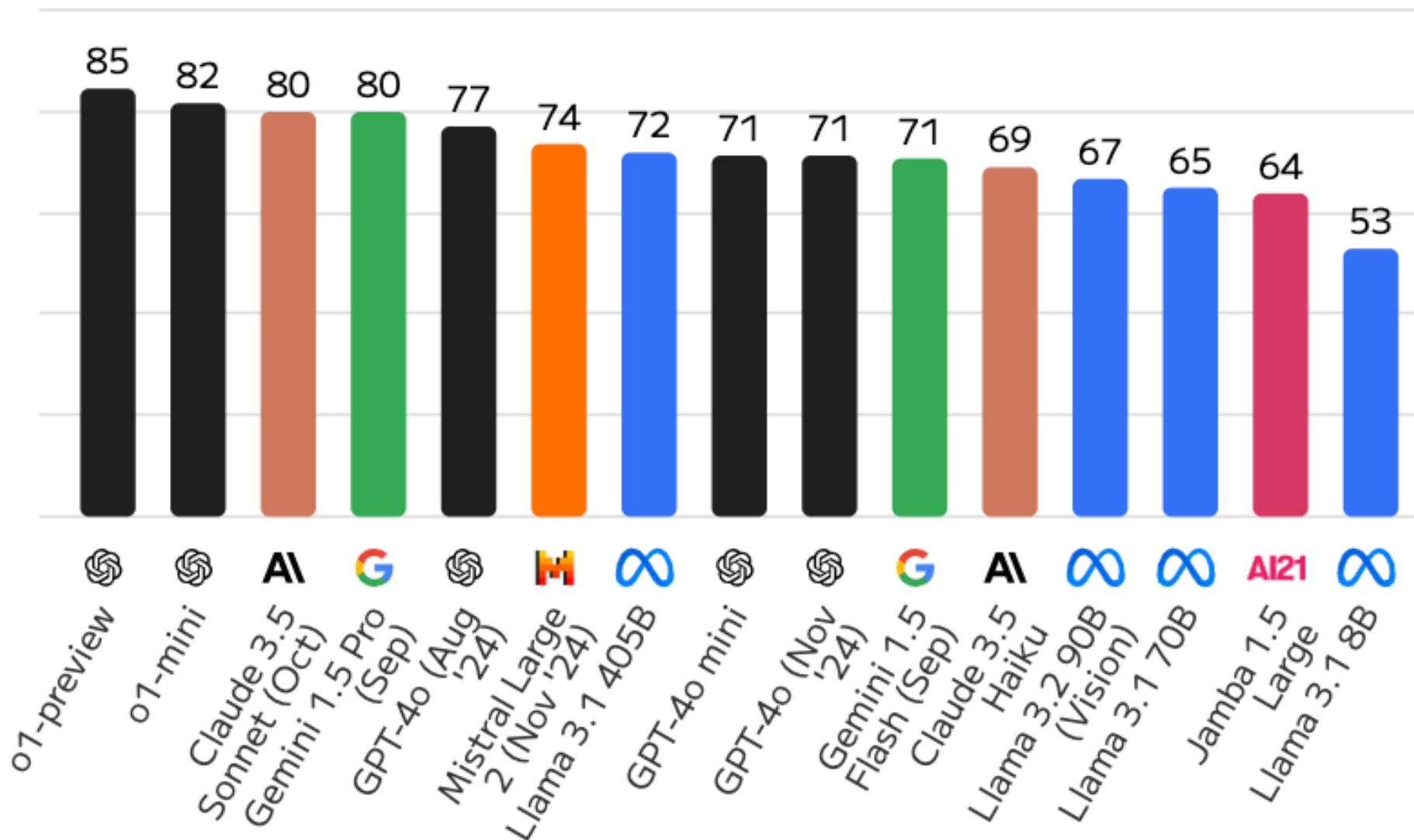
Source: <https://mistral.ai/news/pixtral-large/>

Mistral Pixtral 12B



Large Language Models (LLMs)

Artificial Analysis Quality Index



Large Language Models (LLMs)

Quality vs. Price

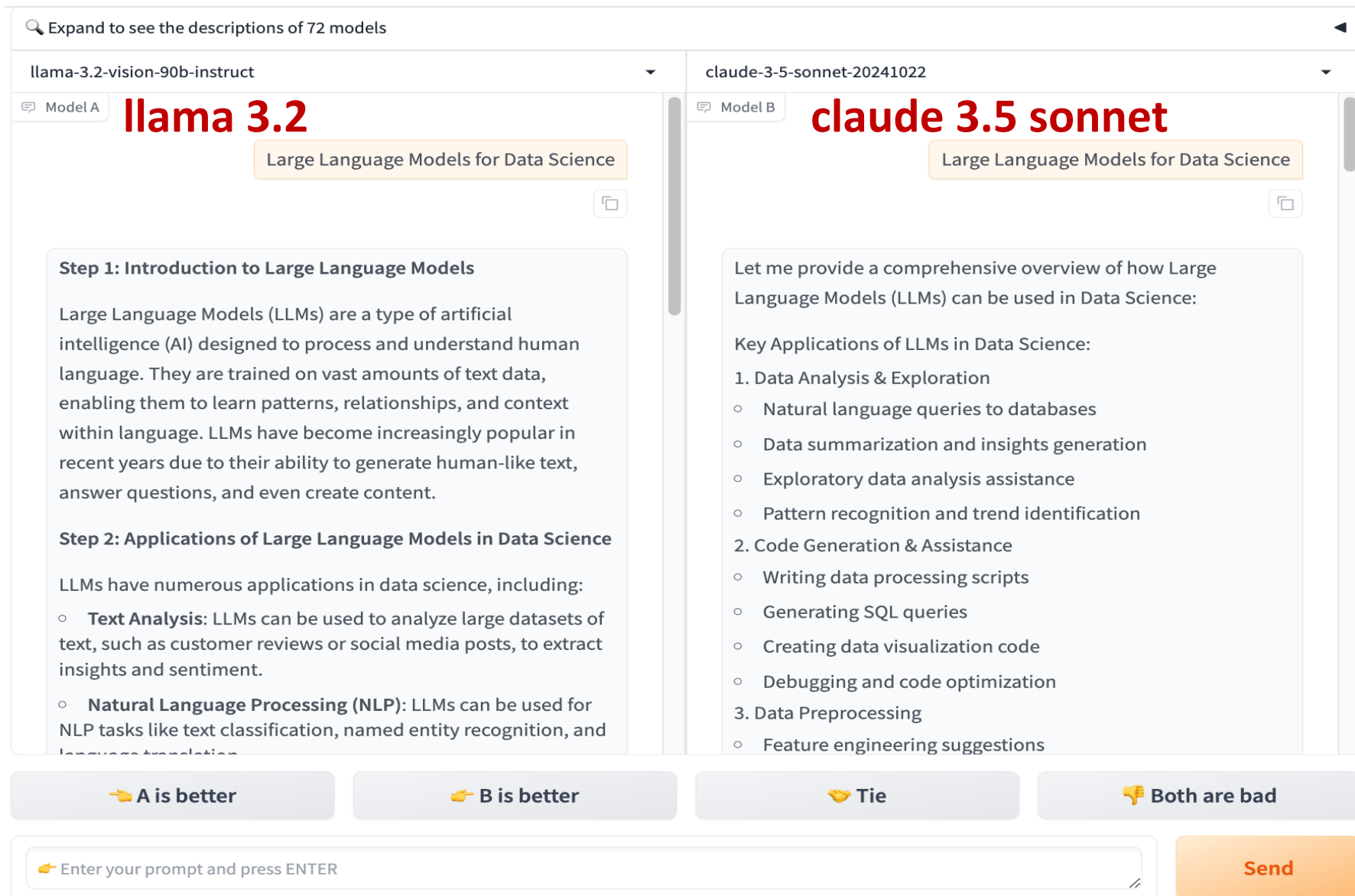
Artificial Analysis Quality Index, Price: USD per 1M Tokens

Most attractive quadrant

- o1-preview
- o1-mini
- GPT-4o (Aug '24)
- GPT-4o mini
- GPT-4o (Nov '24)
- Llama 3.1 405B
- Llama 3.2 90B (Vision)
- Llama 3.1 70B
- Llama 3.1 8B
- Gemini 1.5 Pro (Sep)
- Gemini 1.5 Flash (Sep)
- Claude 3.5 Sonnet (Oct)
- Claude 3.5 Haiku
- Mistral Large 2 (Nov '24)
- Jamba 1.5 Large



Chat with Open Large Language Models: Chatbot Arena



Expand to see the descriptions of 72 models

llama-3.2-vision-90b-instruct Model A **llama 3.2**

claude-3-5-sonnet-20241022 Model B **claude 3.5 sonnet**

Large Language Models for Data Science

Step 1: Introduction to Large Language Models

Large Language Models (LLMs) are a type of artificial intelligence (AI) designed to process and understand human language. They are trained on vast amounts of text data, enabling them to learn patterns, relationships, and context within language. LLMs have become increasingly popular in recent years due to their ability to generate human-like text, answer questions, and even create content.

Step 2: Applications of Large Language Models in Data Science

LLMs have numerous applications in data science, including:

- **Text Analysis:** LLMs can be used to analyze large datasets of text, such as customer reviews or social media posts, to extract insights and sentiment.
- **Natural Language Processing (NLP):** LLMs can be used for NLP tasks like text classification, named entity recognition, and language translation.

Let me provide a comprehensive overview of how Large Language Models (LLMs) can be used in Data Science:

Key Applications of LLMs in Data Science:

1. Data Analysis & Exploration
 - Natural language queries to databases
 - Data summarization and insights generation
 - Exploratory data analysis assistance
 - Pattern recognition and trend identification
2. Code Generation & Assistance
 - Writing data processing scripts
 - Generating SQL queries
 - Creating data visualization code
 - Debugging and code optimization
3. Data Preprocessing
 - Feature engineering suggestions

👉 A is better 👉 B is better 👉 Tie 👉 Both are bad

👉 Enter your prompt and press ENTER Send

<https://lmarena.ai/>

Perplexity.ai



Anonymous Like

Share

New Thread

Home

Discover

Library

Login

Sign Up

Try Pro

Upgrade to Claude-2 or GPT-4, boost Copilot uses, and upload more files.

Learn More

Download

Large Language Models for Data Science

Sources

What Is a Large Language Model and What Can...

snowflake · 1

What are Large Language Models? | NVIDIA

nvidia · 2

Catch Up On Large Language Models | by...

towardsdatascien... · 3



View 3 more

Answer

Large Language Models (LLMs) are advanced AI systems designed to understand human language intricacies and generate intelligent, creative responses to queries ¹. They are deep learning algorithms that can recognize, summarize, translate, predict, and generate content using very large datasets ². LLMs are trained on enormous data sets typically measured in petabytes, sourced from books, articles, websites, and other text-based sources ¹. They are dynamic tools that are continuously updated and refined as new data becomes available ¹.

In the field of data science, LLMs can potentially transform how teams source, manage, and analyze data ¹. They are used in numerous data science applications, and their ability to process and interpret vast amounts of text data has made them an indispensable part of

Ask follow-up...

Copilot



social media posts or customer reviews, to determine whether the overall sentiment is

<https://www.perplexity.ai/>

Generative AI (Gen AI)

AI Generated Content (AIGC)

Image Generation

Instruction 1:

An astronaut riding a horse in a photorealistic style.

Instruction 2:

Teddy bears working on new AI research on the moon in the 1980s.

Figure 1



Figure 2

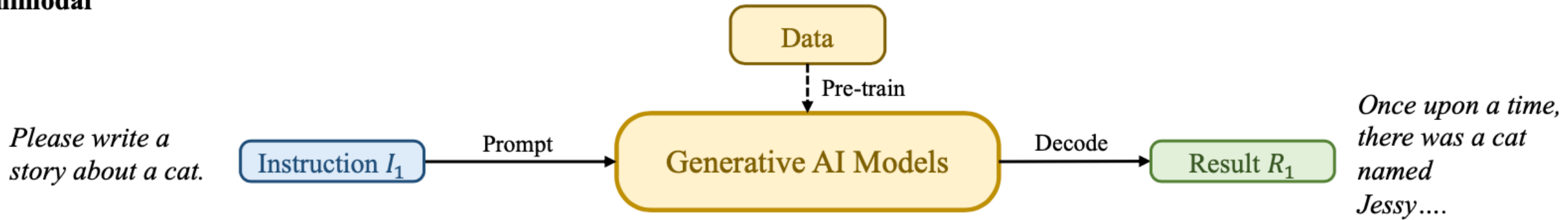


 **OpenAI DALL·E 2**

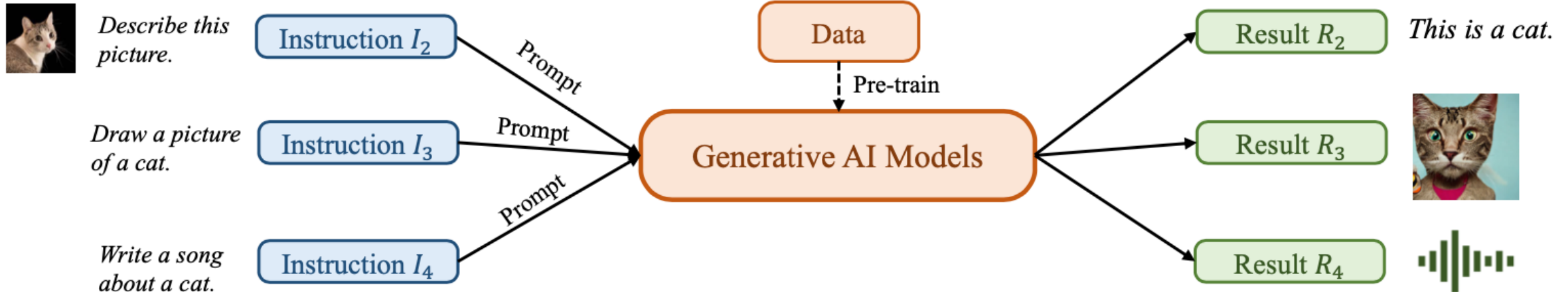
Generative AI (Gen AI)

AI Generated Content (AIGC)

Unimodal

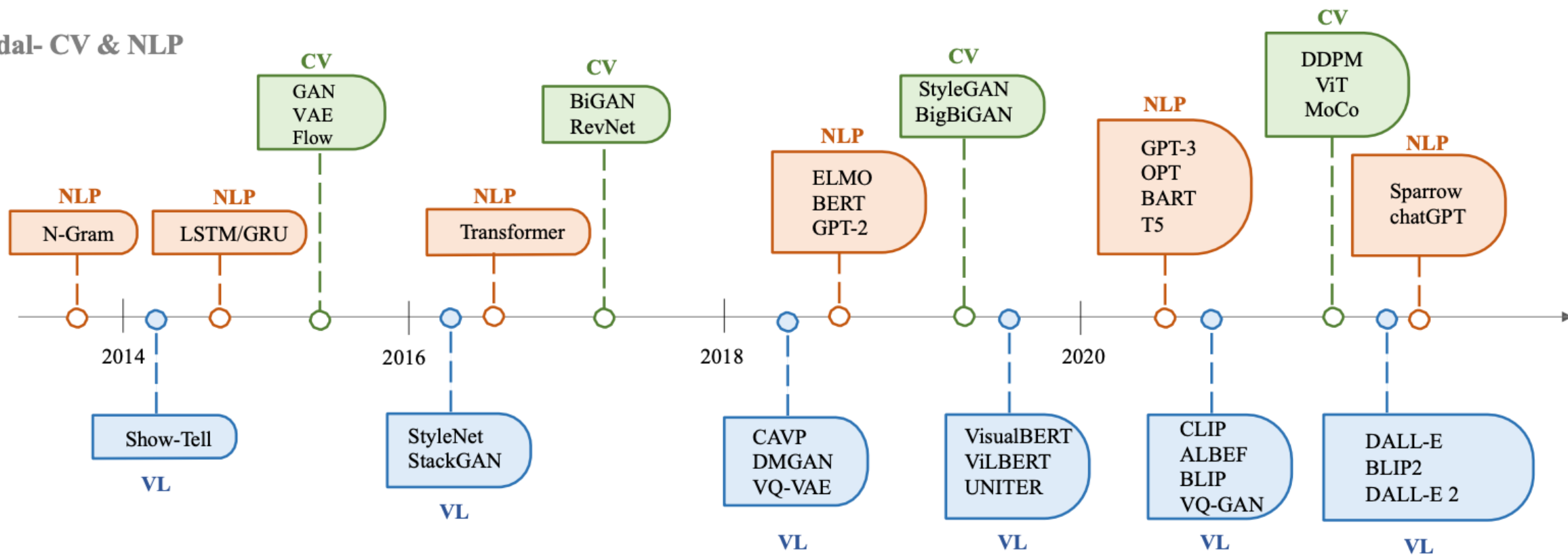


Multimodal



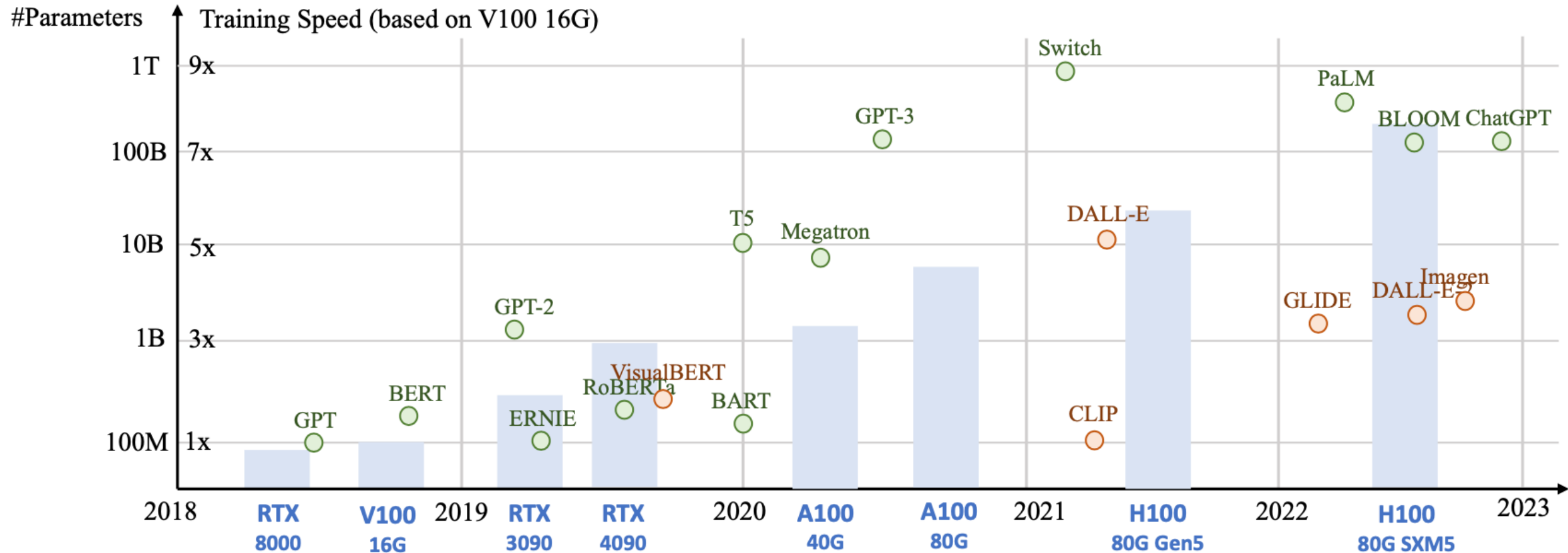
The history of Generative AI in CV, NLP and VL

Unimodal- CV & NLP

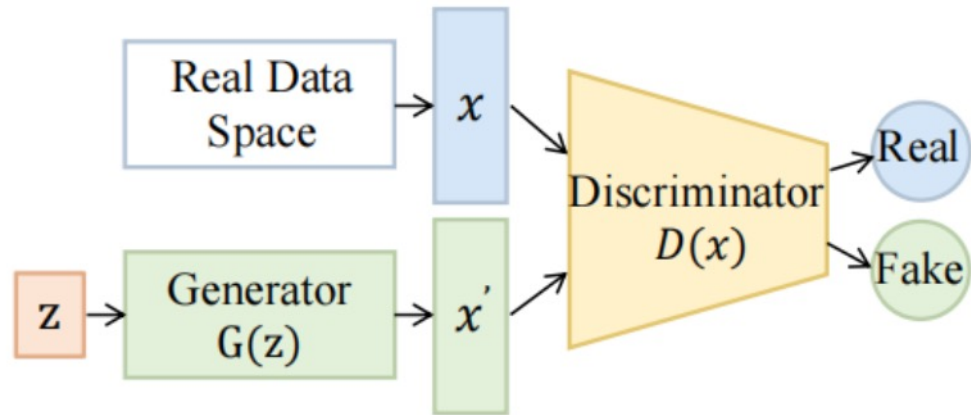


Multimodal – Vision Language

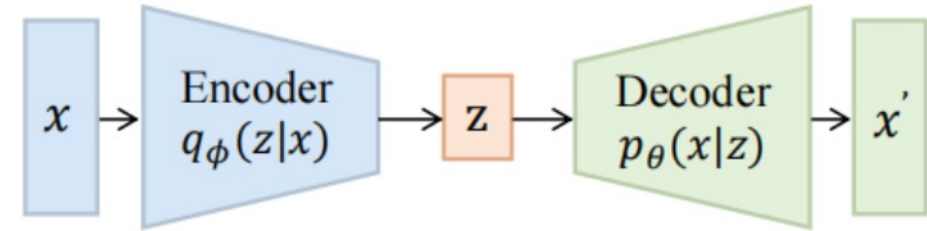
Generative AI Foundation Models



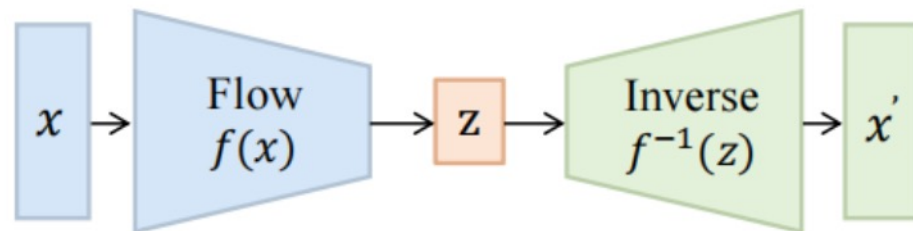
Categories of Vision Generative Models



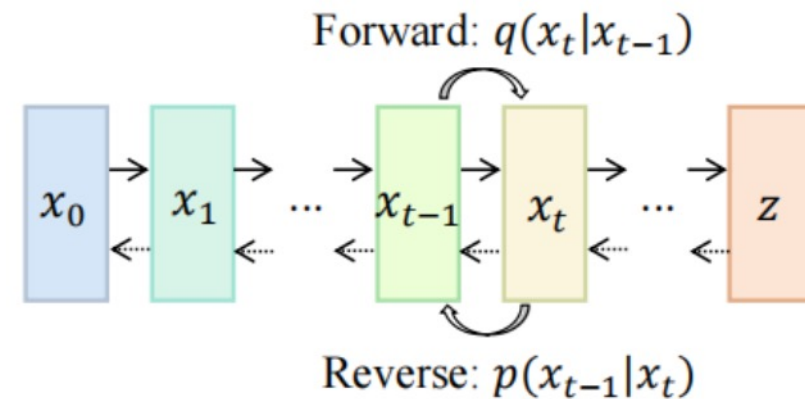
(1) Generative adversarial networks



(2) Variational autoencoders

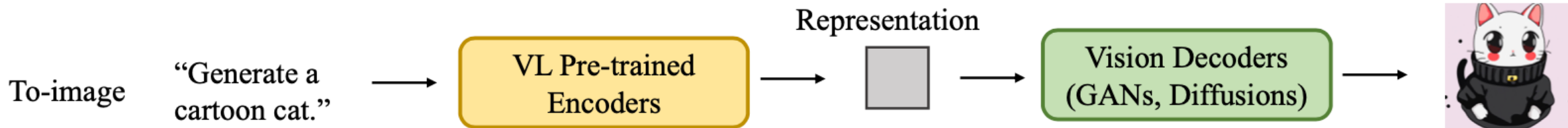
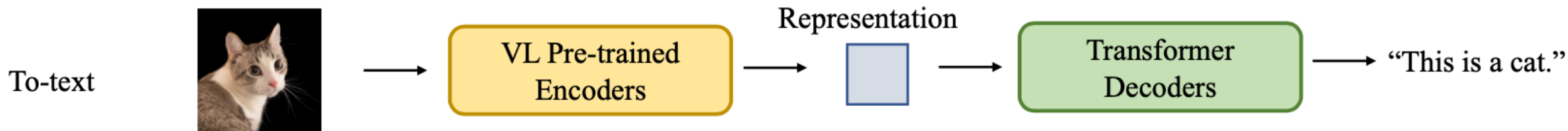
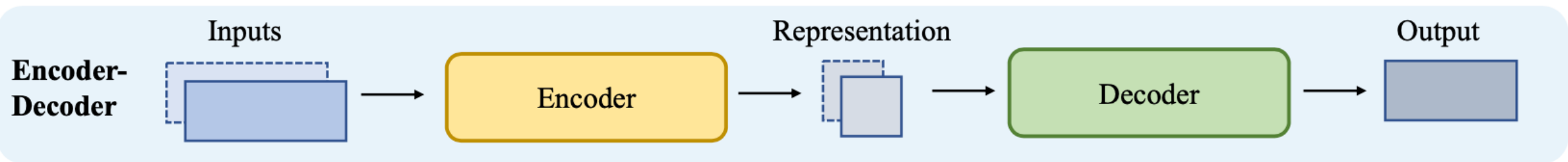


(3) Normalizing flows

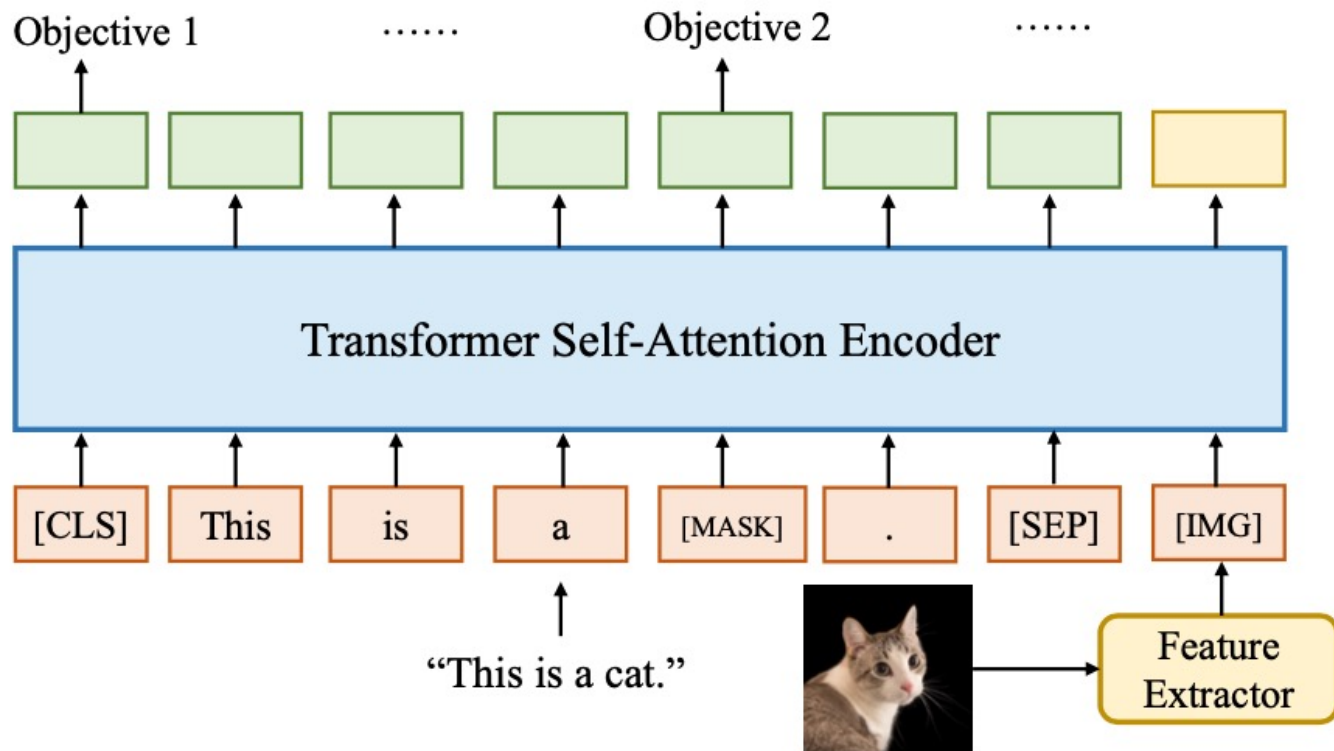


(4) Diffusion models

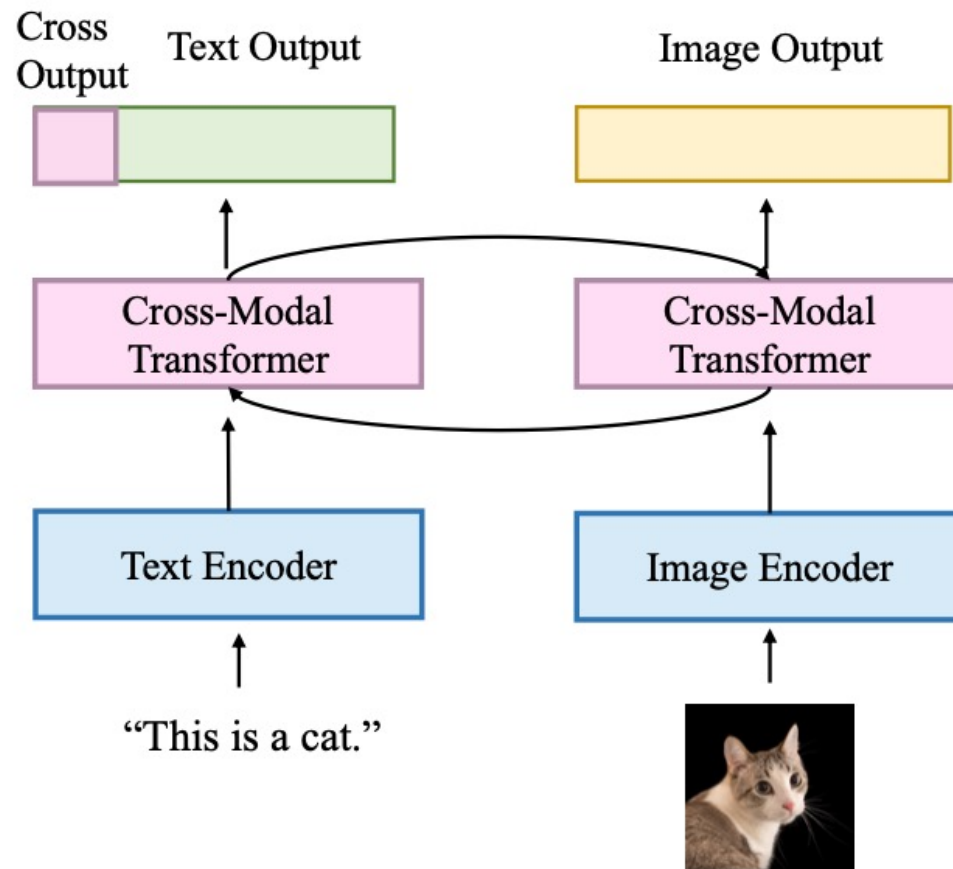
The General Structure of Generative Vision Language



Two Types of Vision Language Encoders: Concatenated Encoders and Cross-aligned Encoders

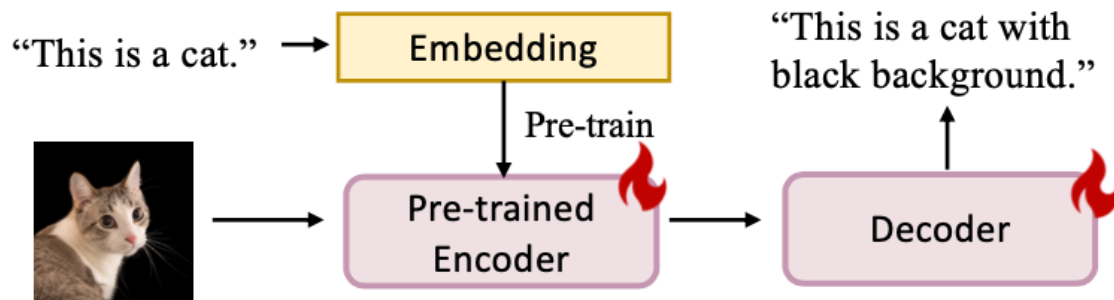


(a) Concatenated Encoder

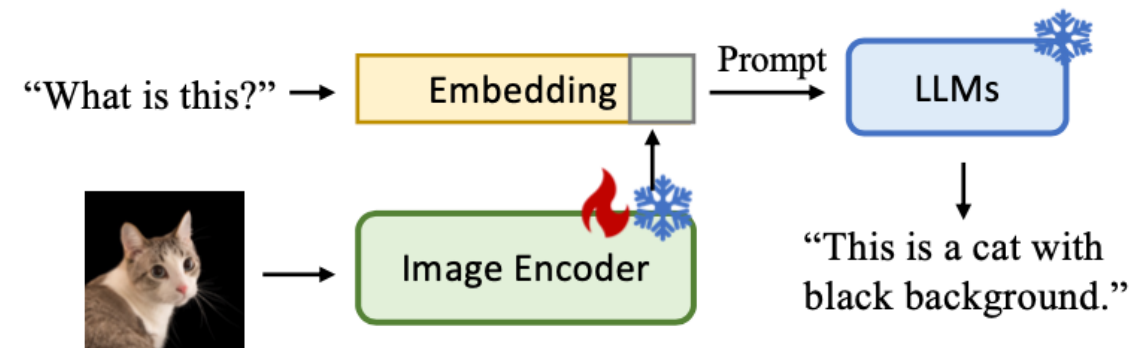


(b) Cross-aligned Encoder

Two Types of to-language Decoder Models: Jointly-trained Models and Frozen Models

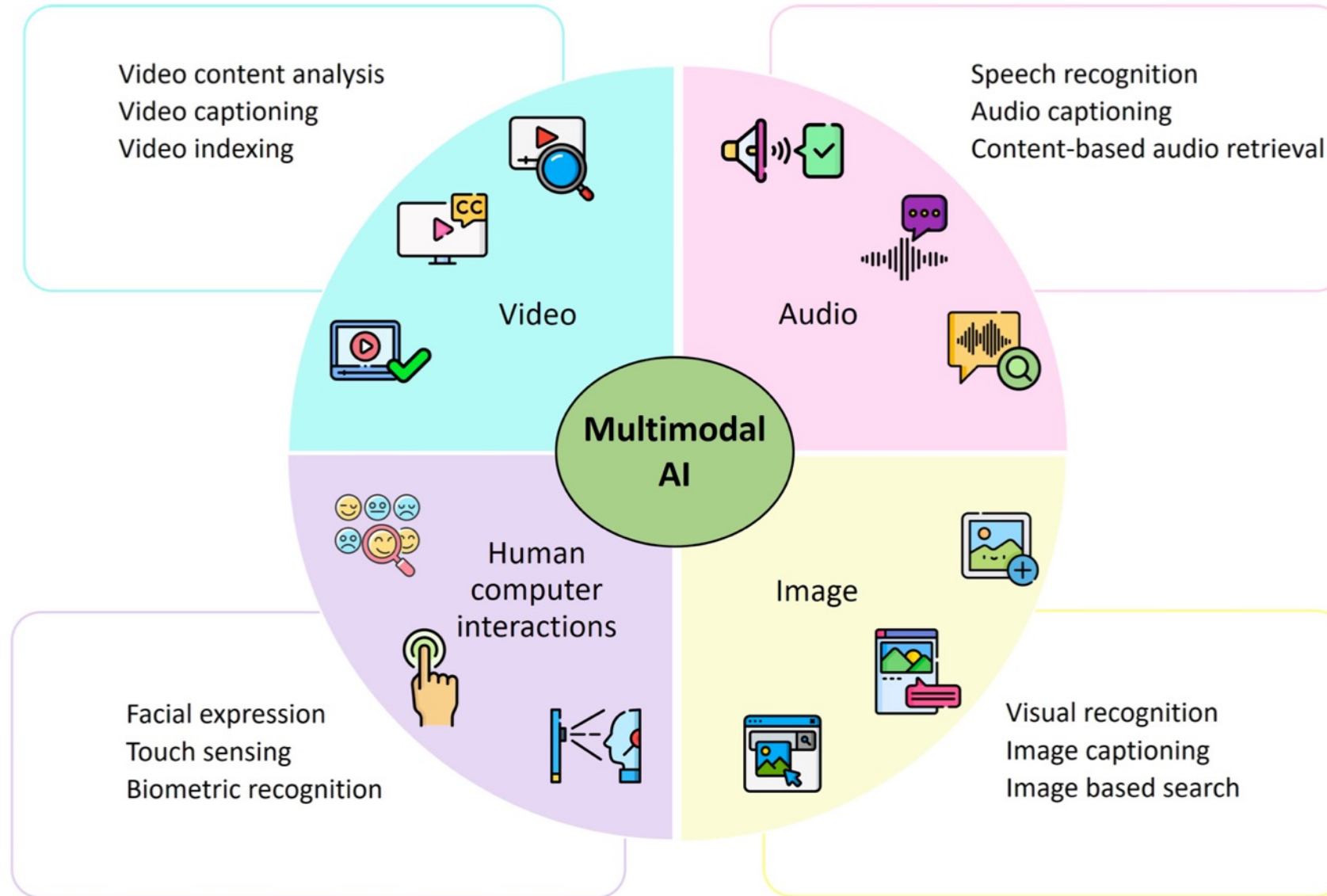


(a) Jointly-trained Models



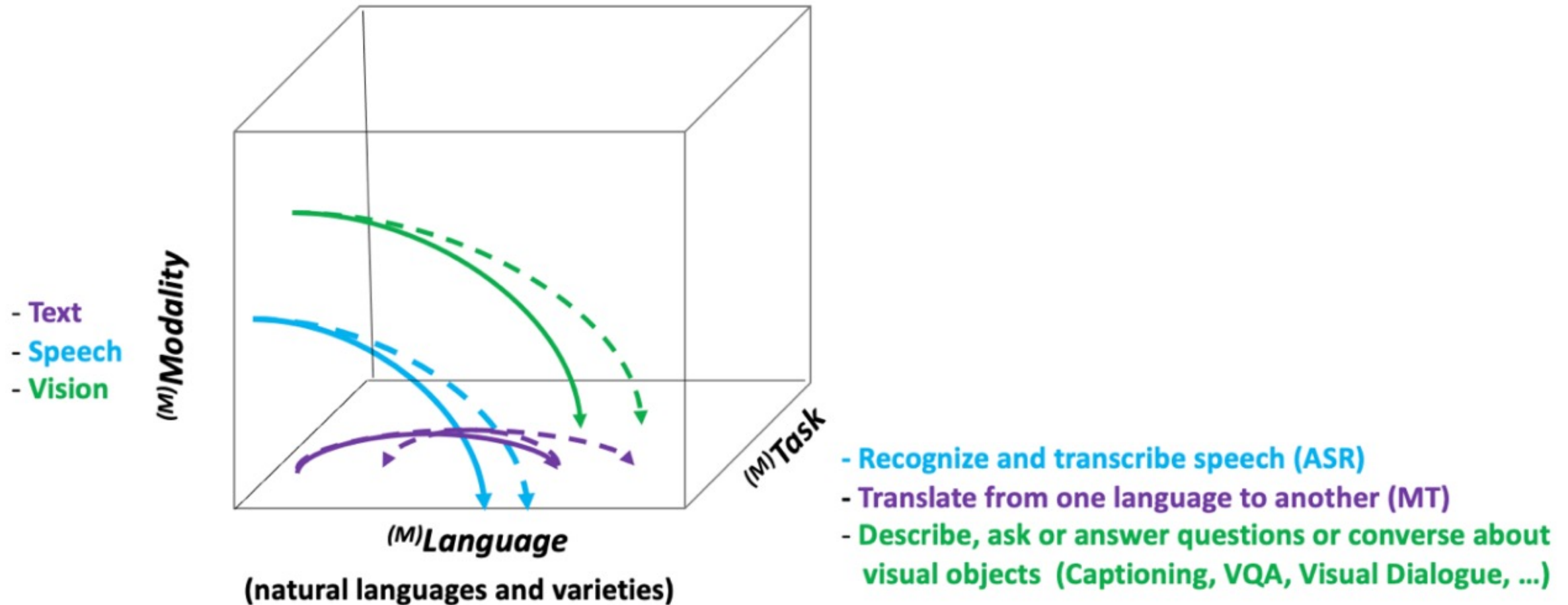
(b) Frozen Models

Technological Integration for Multimodal AI



NLG from a Multilingual, Multimodal and Multi-task perspective

Multi³(Natural Language) Generation

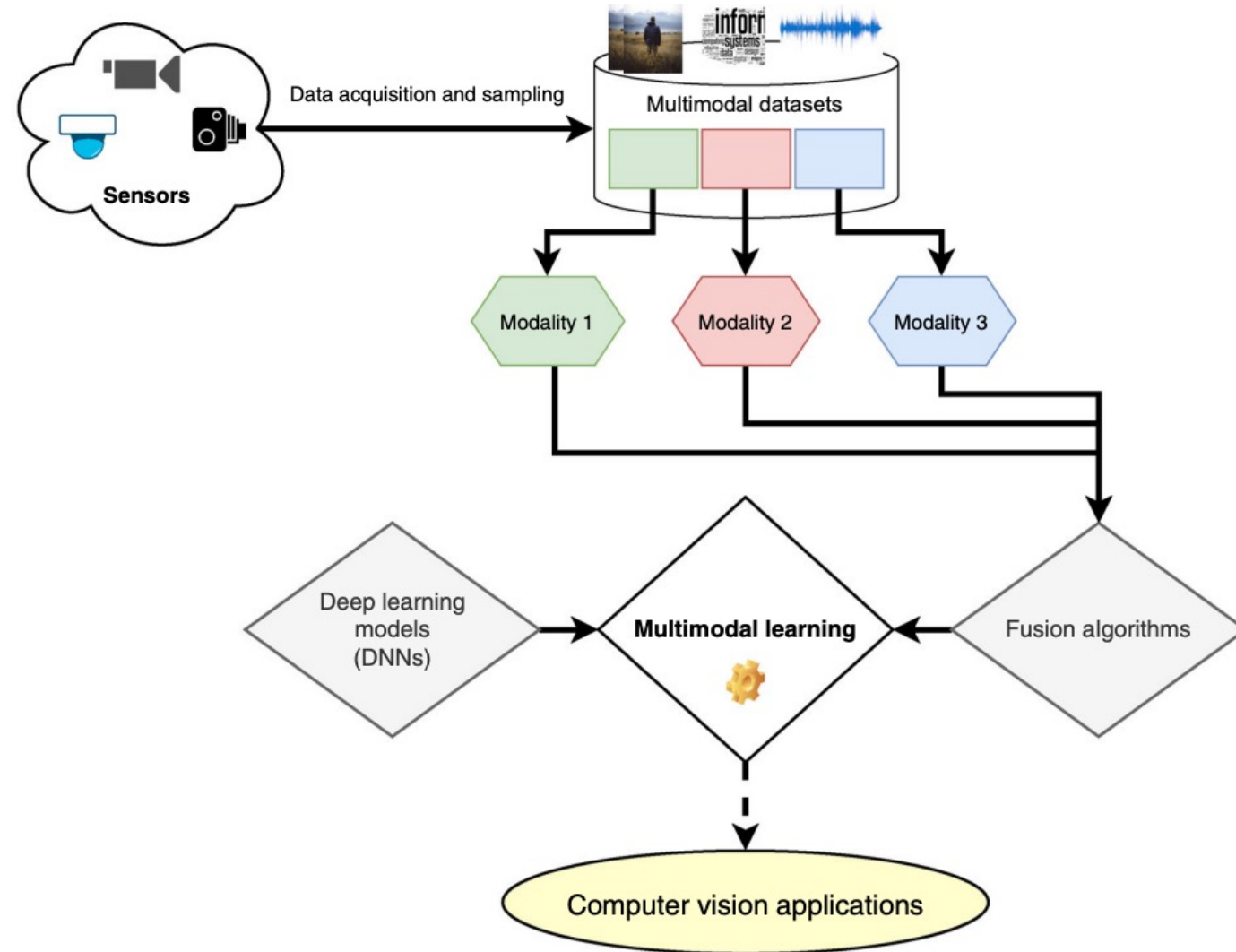


Source: Erdem, Erkut, Menekse Kuyu, Semih Yagcioglu, Anette Frank, Letitia Parcalabescu, Barbara Plank, Andrii Babii et al.

"Neural Natural Language Generation: A Survey on Multilinguality, Multimodality, Controllability and Learning." Journal of Artificial Intelligence Research 73 (2022): 1131-1207.

Multimodal Pipeline

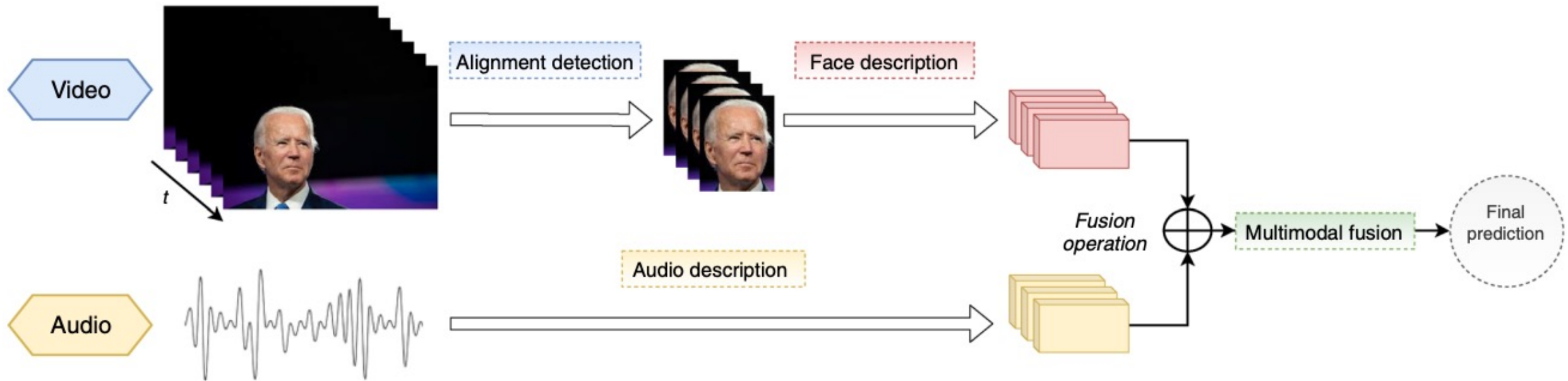
that includes three different modalities (Image, Text, Audio)



Source: Bayoudh, Khaled, Raja Knani, Fayçal Hamdaoui, and Abdellatif Mtibaa (2022).

"A survey on deep multimodal learning for computer vision: advances, trends, applications, and datasets." The Visual Computer 38, no. 8: 2939-2970.

Video and Audio Multimodal Fusion



Source: Bayoudh, Khaled, Raja Knani, Fayçal Hamdaoui, and Abdellatif Mtibaa (2022).

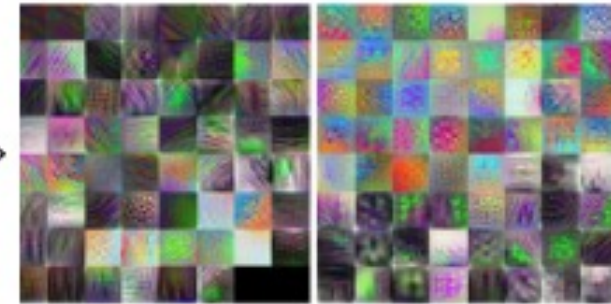
"A survey on deep multimodal learning for computer vision: advances, trends, applications, and datasets." The Visual Computer 38, no. 8: 2939-2970.

Visual and Textual Representation

Image



Visual representations (Dense)



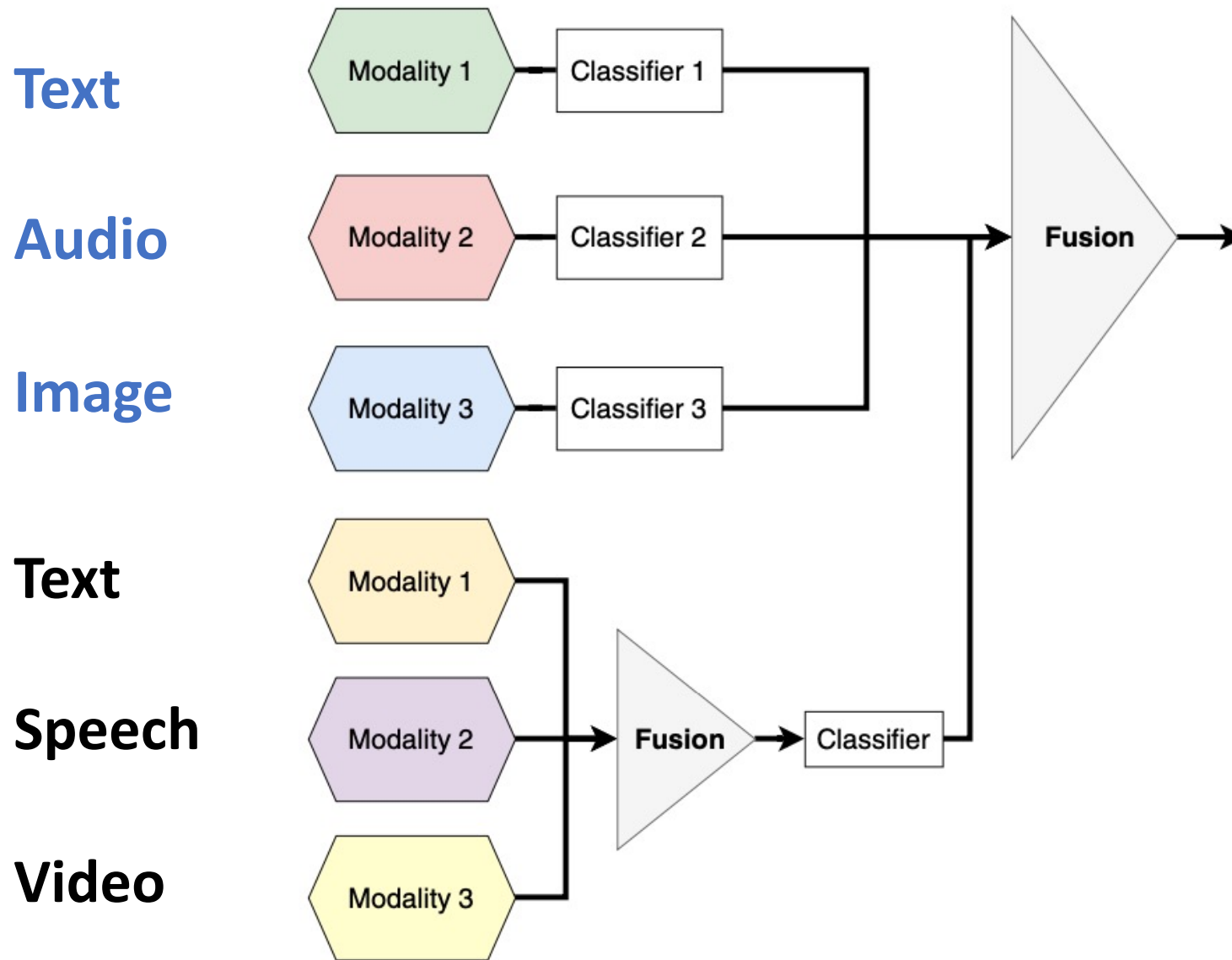
Text

This is the oldest and most important defensive work to have been built along the North African coastline by the Arab conquerors in the early days of Islam. Founded in 796, this building underwent several modifications during the medieval period. Initially, it formed a quadrilateral and then was composed of four buildings giving onto two inner courtyards.

Textual representations (Sparse)



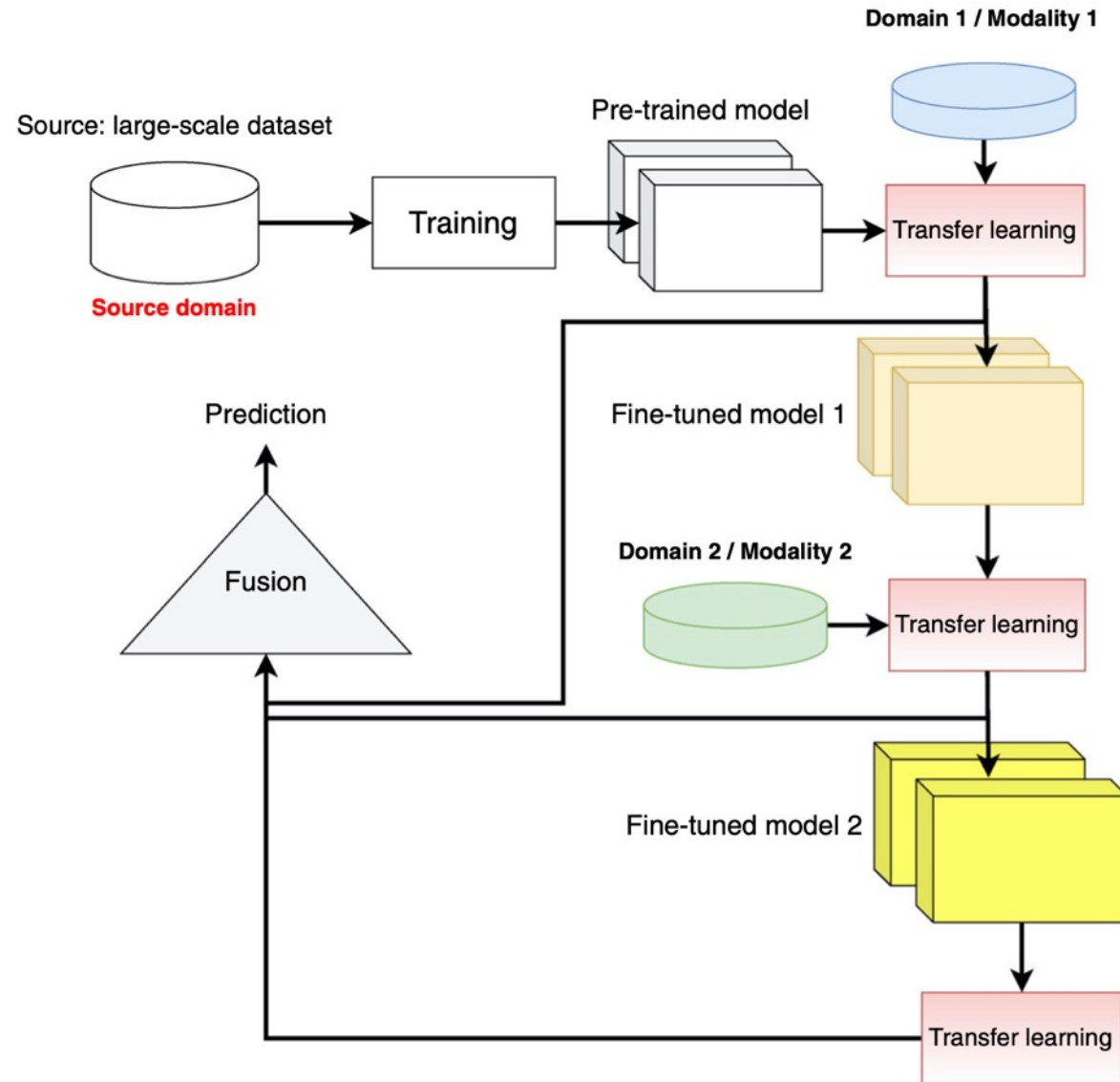
Hybrid Multimodal Data Fusion



Source: Bayouadh, Khaled, Raja Knani, Fayçal Hamdaoui, and Abdellatif Mtibaa (2022).

"A survey on deep multimodal learning for computer vision: advances, trends, applications, and datasets." The Visual Computer 38, no. 8: 2939-2970.

Multimodal Transfer Learning

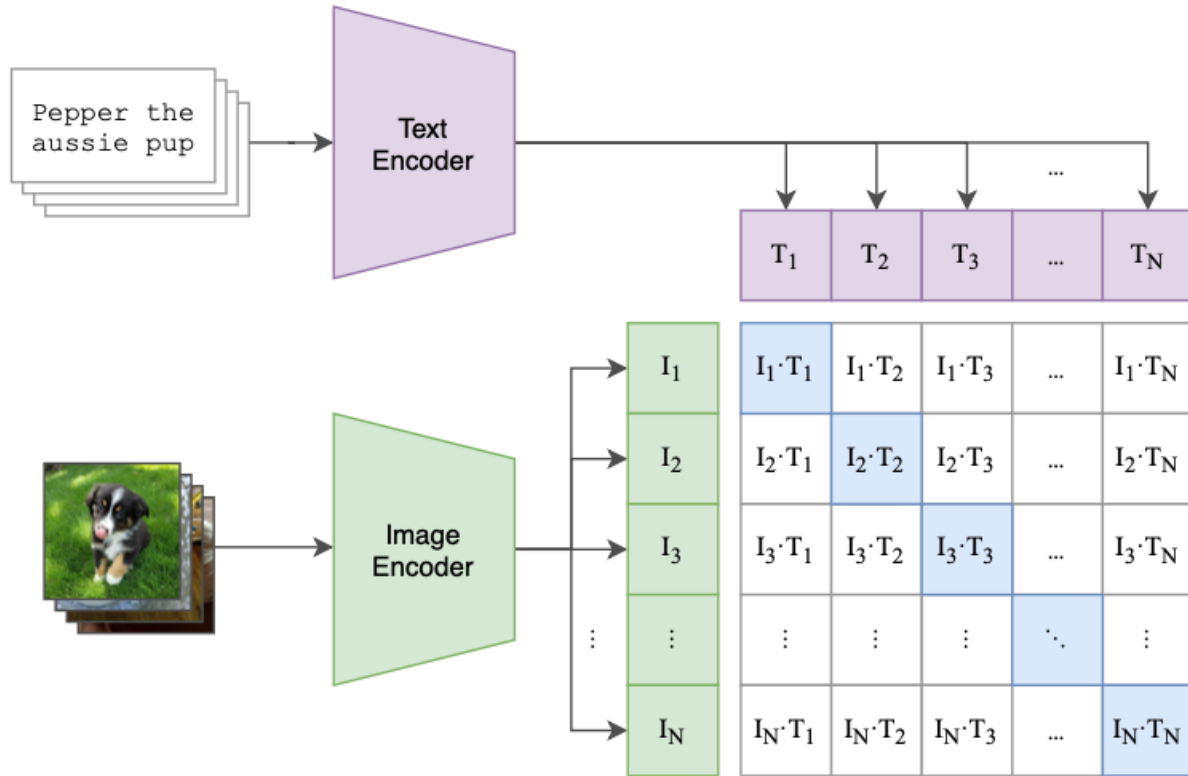


Source: Bayoudh, Khaled, Raja Knani, Fayçal Hamdaoui, and Abdellatif Mtibaa (2022).

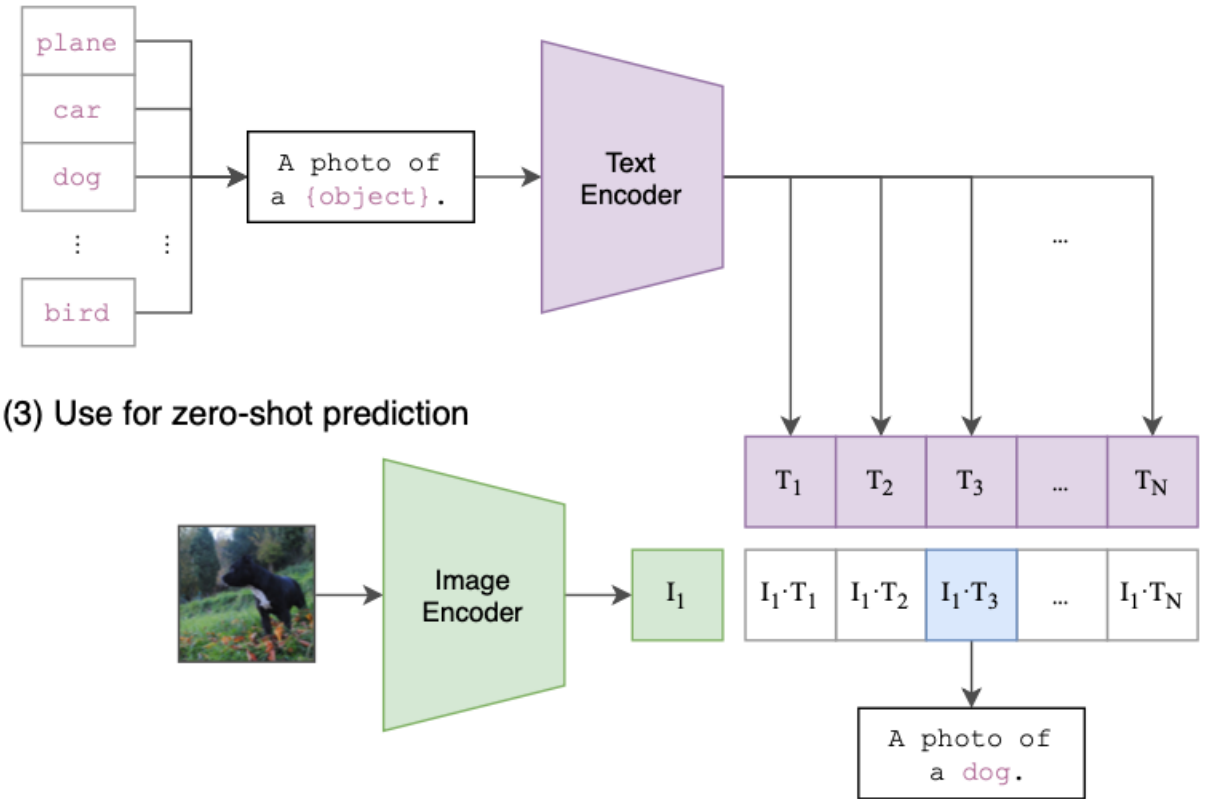
"A survey on deep multimodal learning for computer vision: advances, trends, applications, and datasets." The Visual Computer 38, no. 8: 2939-2970.

CLIP: Learning Transferable Visual Models From Natural Language Supervision

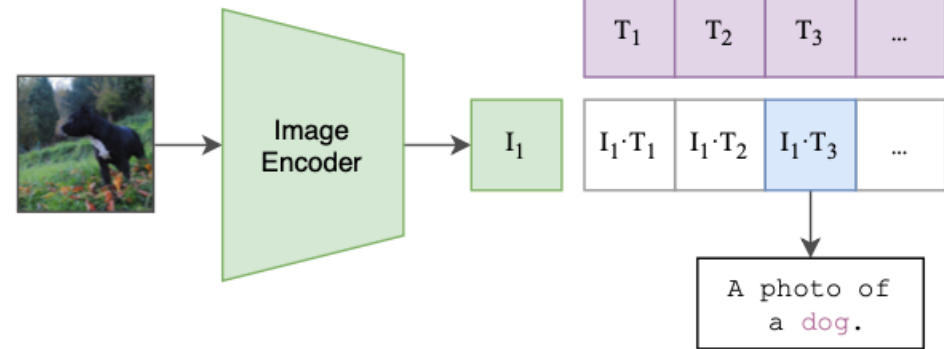
(1) Contrastive pre-training



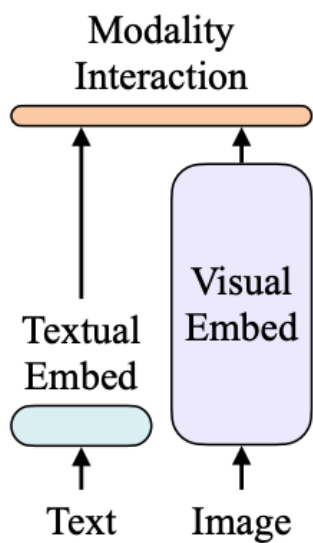
(2) Create dataset classifier from label text



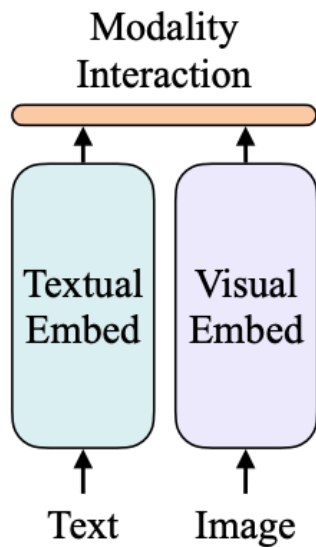
(3) Use for zero-shot prediction



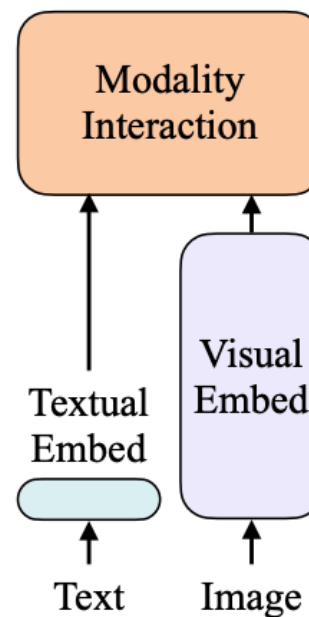
ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision



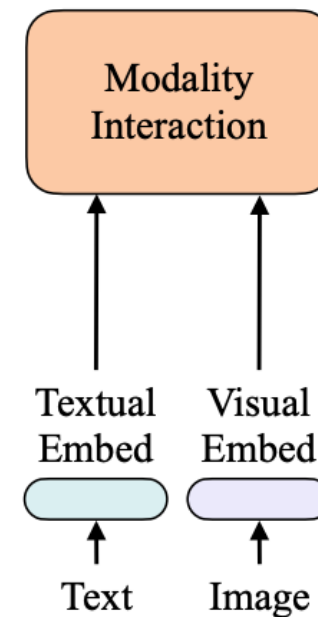
(a) $VE > TE > MI$



(b) $VE = TE > MI$



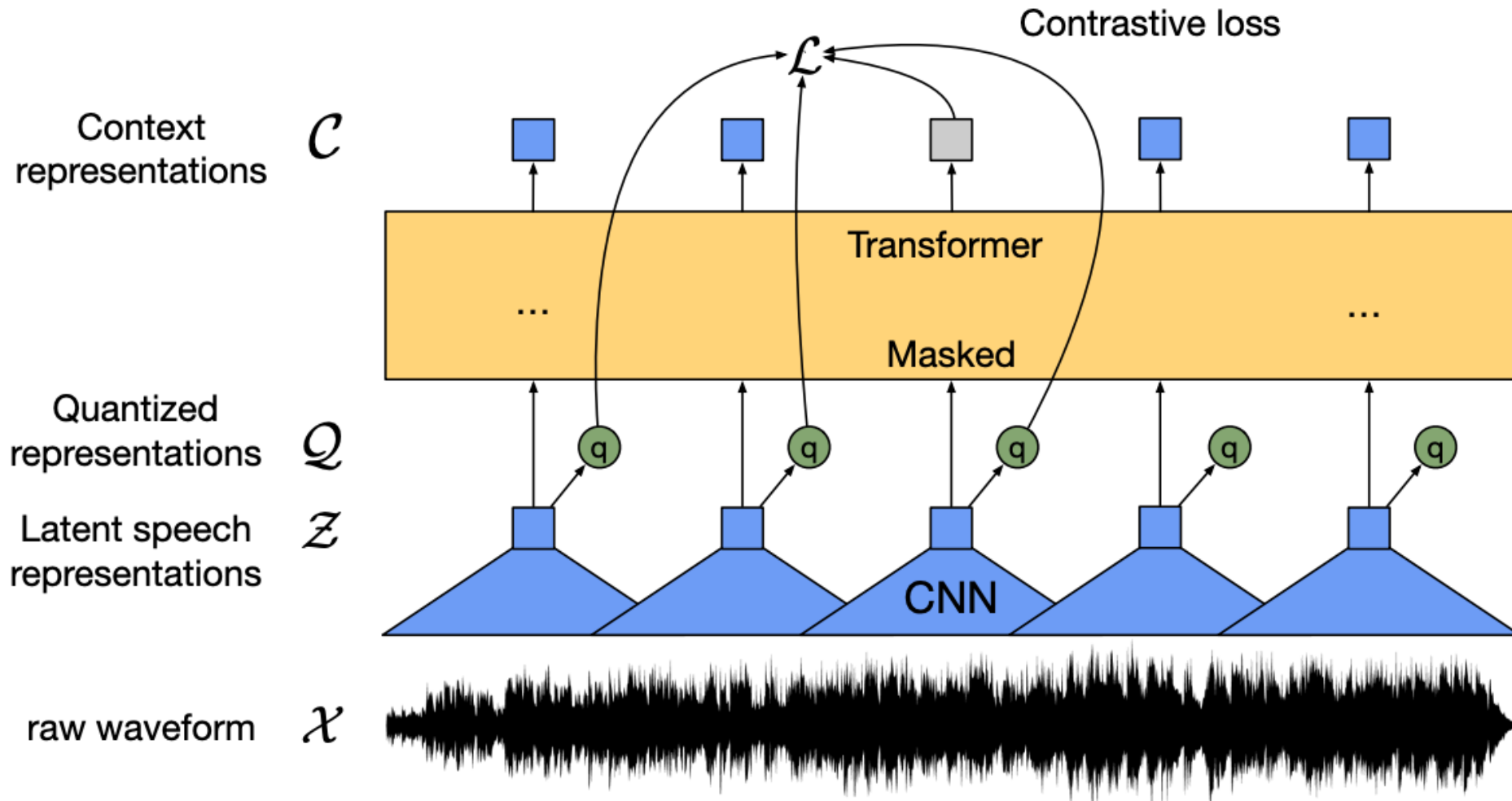
(c) $VE > MI > TE$



(d) $MI > VE = TE$

wav2vec 2.0:

A framework for self-supervised learning of speech representations

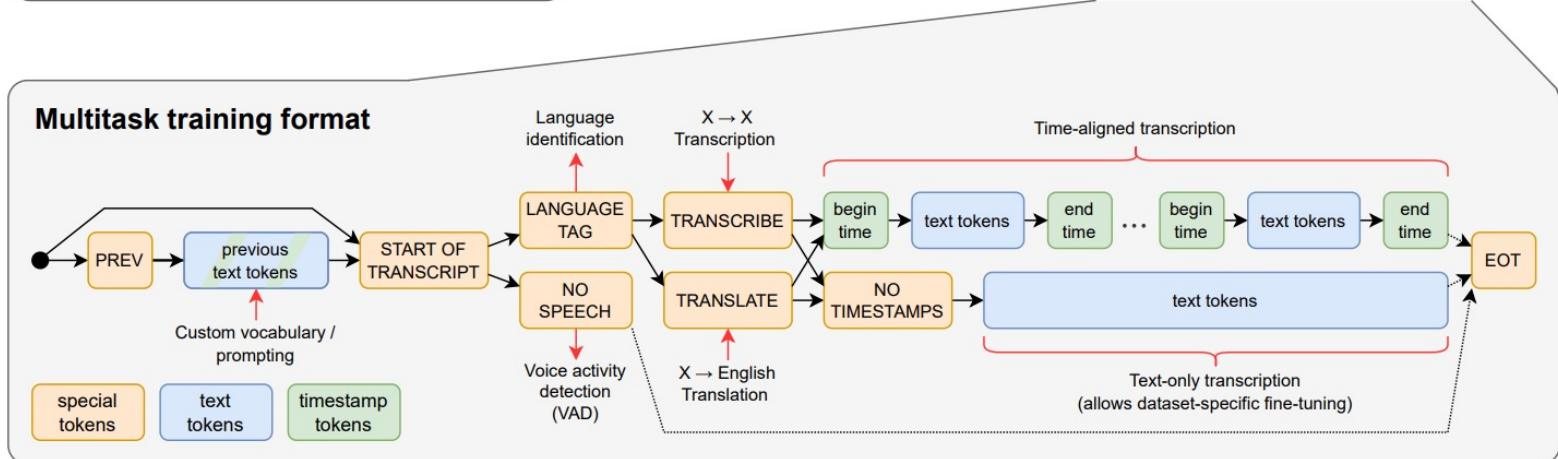
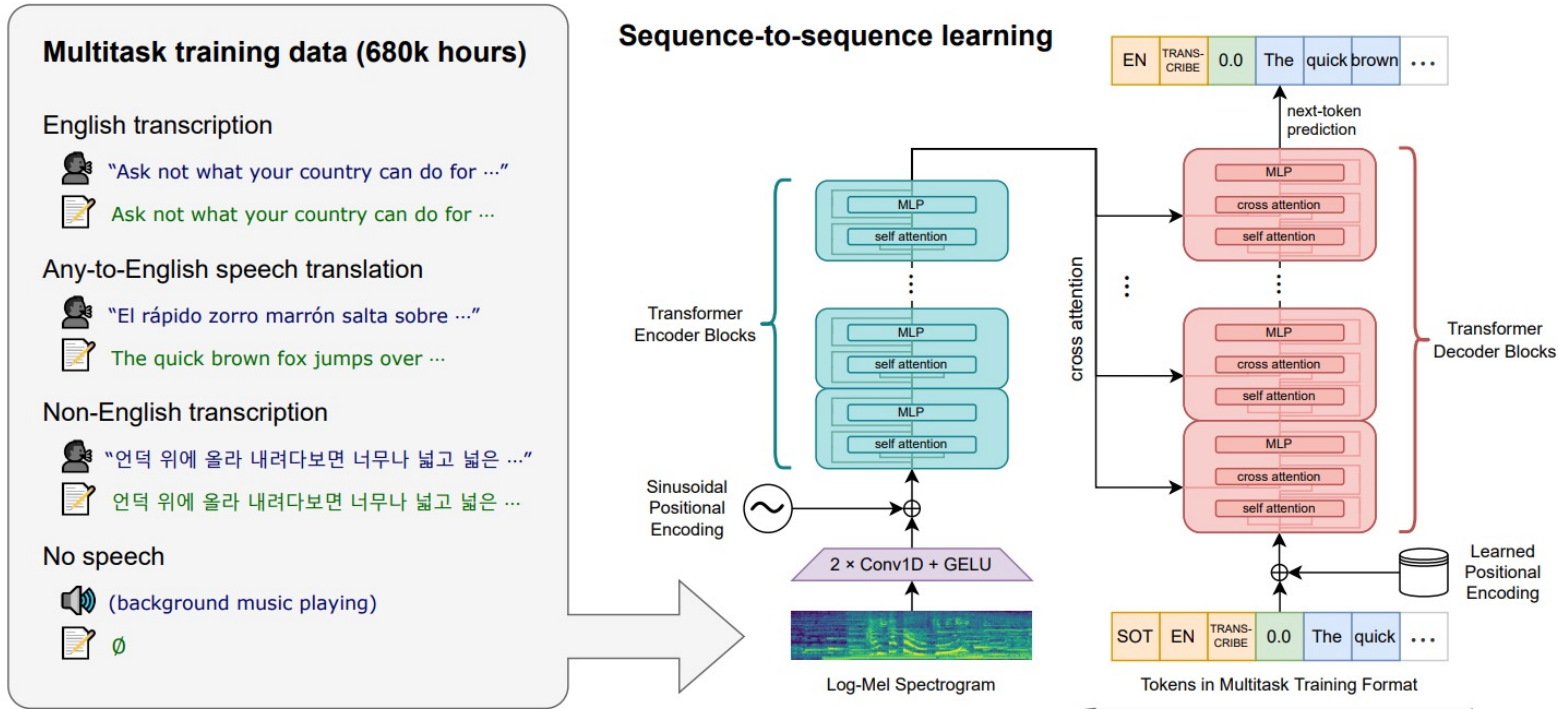


Source: Baevski, Alexei, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli.

"wav2vec 2.0: A framework for self-supervised learning of speech representations." Advances in Neural Information Processing Systems 33 (2020): 12449-12460.

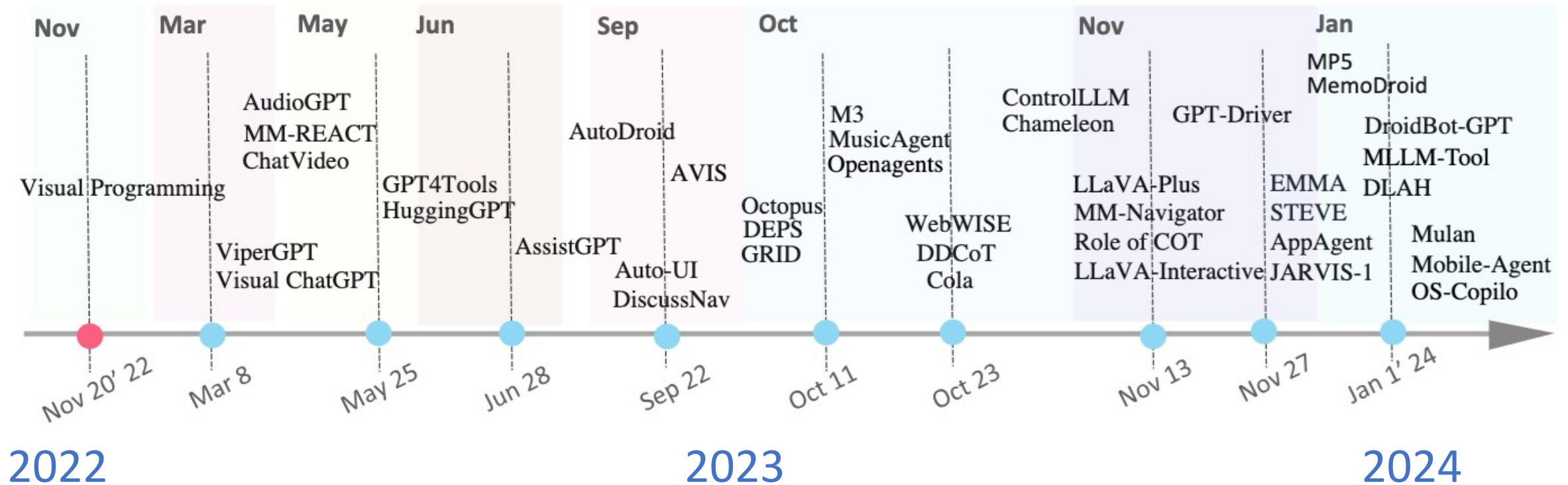
Whisper:

Robust Speech Recognition via Large-Scale Weak Supervision

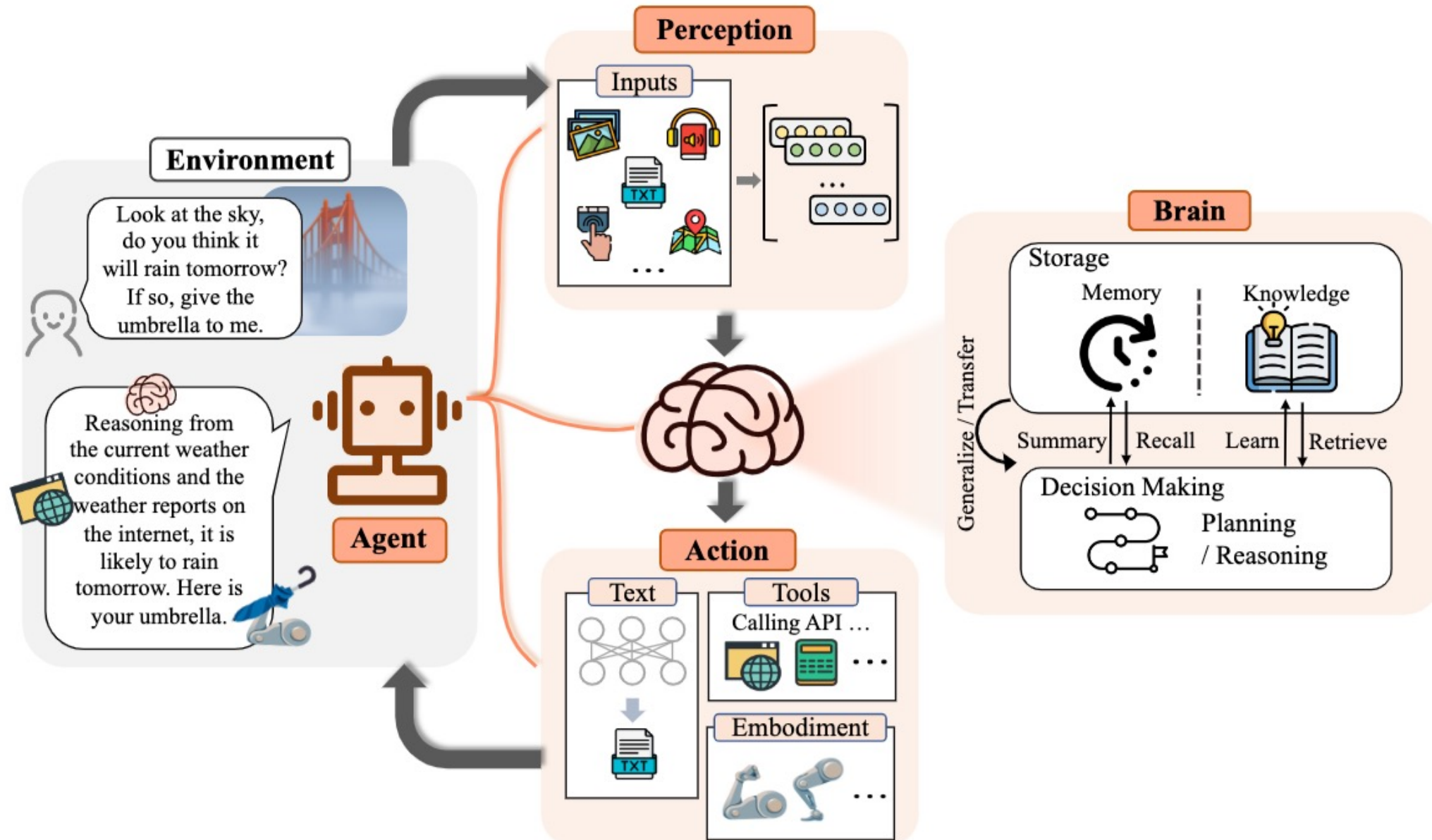


LLM-powered Multimodal Agents

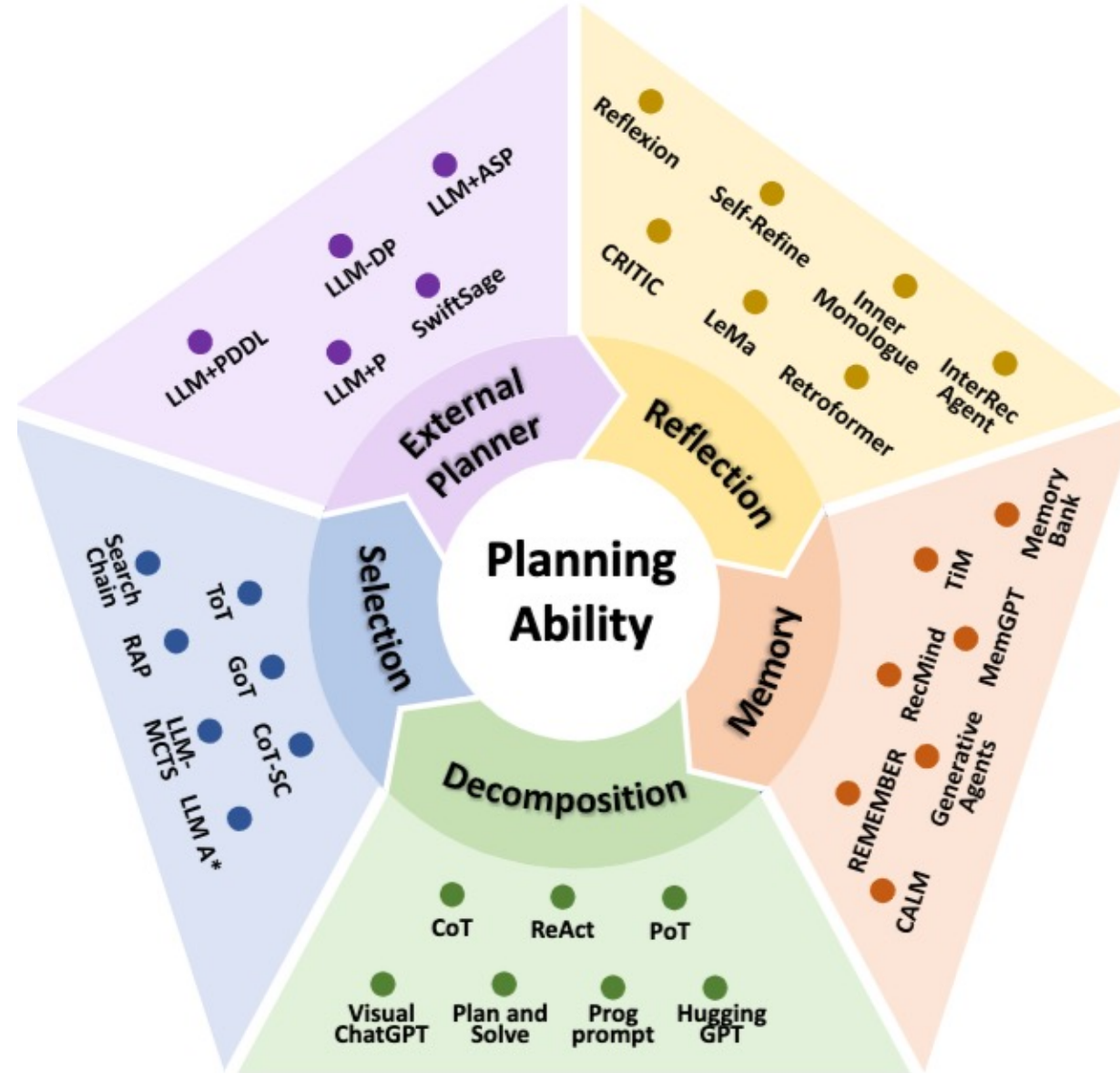
Large Multimodal Agents (LMAs)



Large Language Model (LLM) Based Agents



Taxonomy on LLM-Agent planning

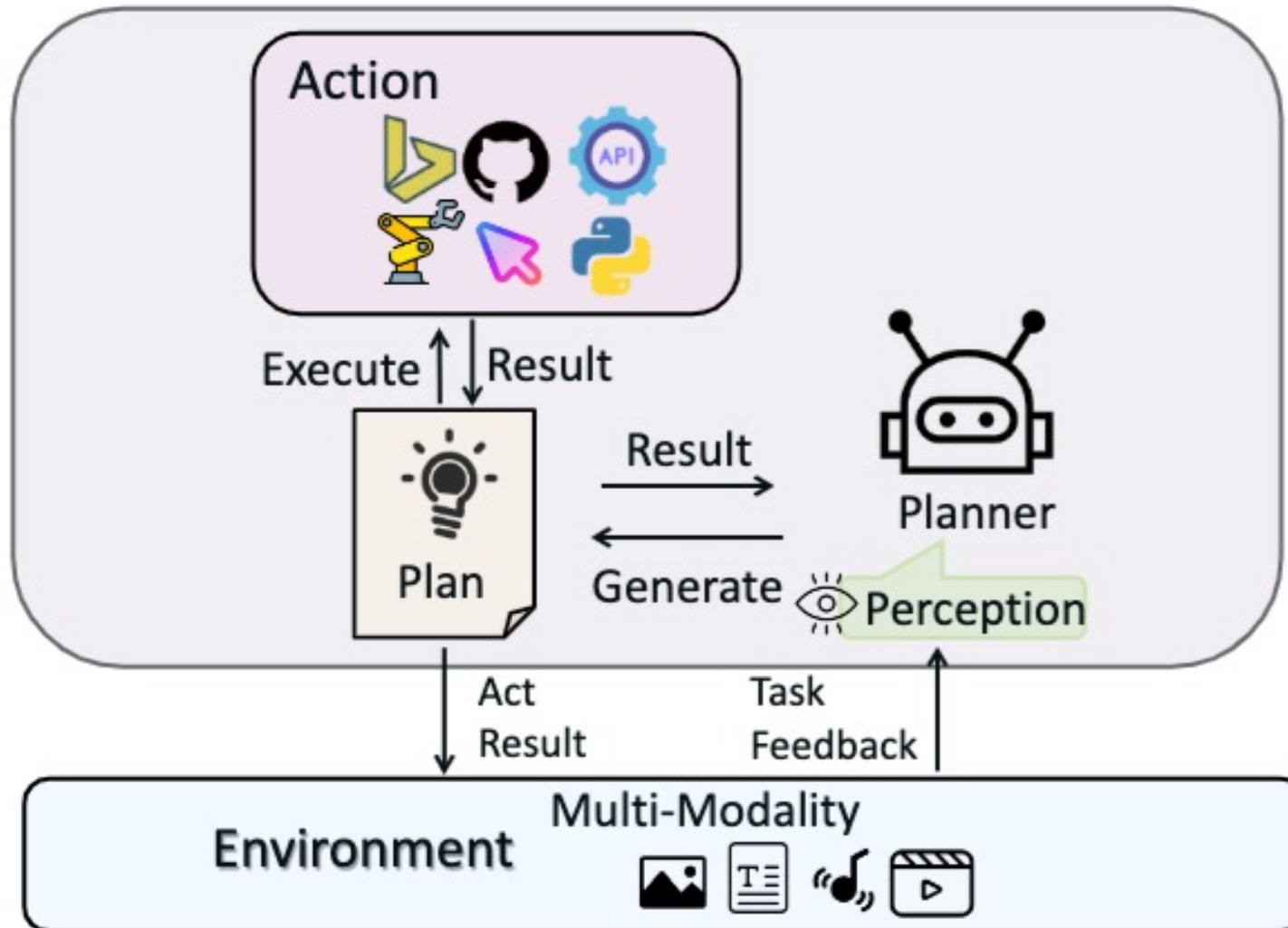


Taxonomy on LLM-Agent planning

Method	Idea	LLM's task	Formulation	Representative works
Task Decomposition	Divide and Conquer	Task decomposition Subtask planning	$[g_i] = \text{decompose}(E, g; \Theta, \mathcal{P});$ $p^i = \text{sub-plan}(E, g_i; \Theta, \mathcal{P})$	CoT [2022], ReAct [2022], HuggingGPT [2023]
Multi-plan Selection	Generate multiple plans and select the optimal	Plans generation Plans evaluation	$P = \text{plan}(E, g; \Theta, \mathcal{P});$ $p^* = \text{select}(E, g, P; \Theta, \mathcal{F})$	ToT [2023], GoT [2023], CoT-SC [2022b]
External Planner-aided	Formalize tasks and utilize external planner	Task formalization	$h = \text{formalize}(E, g; \Theta, \mathcal{P});$ $p = \text{plan}(E, g, h; \Phi)$	LLM+P [2023a], LLM+PDDL [2023]
Reflection & Refinement	Reflect on experiences and refine plans	Plan generation Reflection Refinement	$p_0 = \text{plan}(E, g; \Theta, \mathcal{P});$ $r_i = \text{reflect}(E, g, p_i; \Theta, \mathcal{P});$ $p_{i+1} = \text{refine}(E, g, p_i, r_i; \Theta, \mathcal{P})$	Reflexion [2023], CRITIC [2023], Self-Refine [2023]
Memory-aided Planning	Leverage memory to aid planning	Plan generation Memory extraction	$m = \text{retrieve}(E, g; \mathcal{M});$ $p = \text{plan}(E, g, m; \Theta, \mathcal{P})$	REMEMBER [2023a], MemoryBank [2023]

Large Multimodal Agents (LMAs)

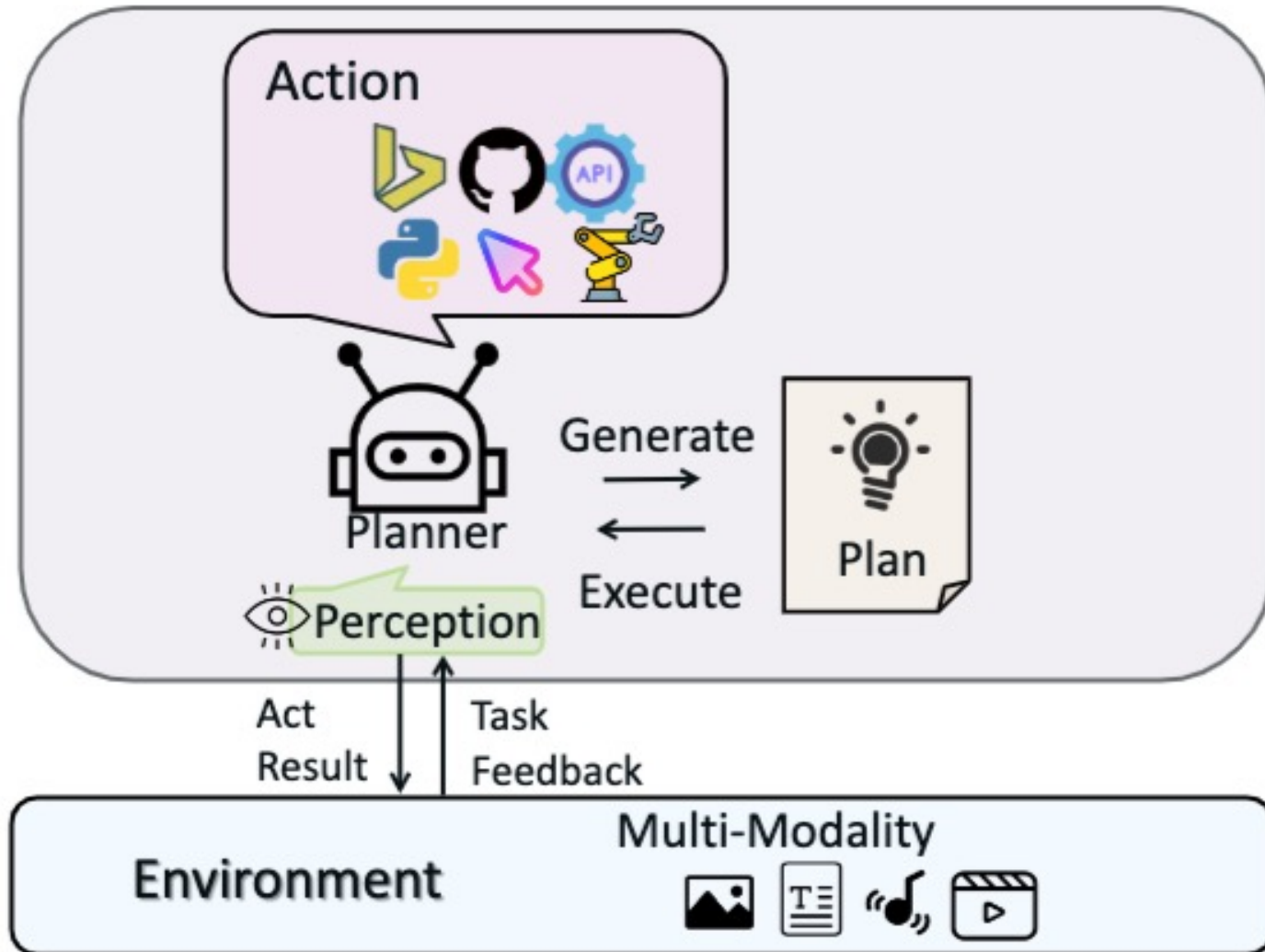
(a) Type I: **Closed-source LLMs as Planners** w/o Longterm Memory.



Use prompt techniques to guide closed-source LLMs in decision-making and planning to complete tasks without long memory.

Large Multimodal Agents (LMAs)

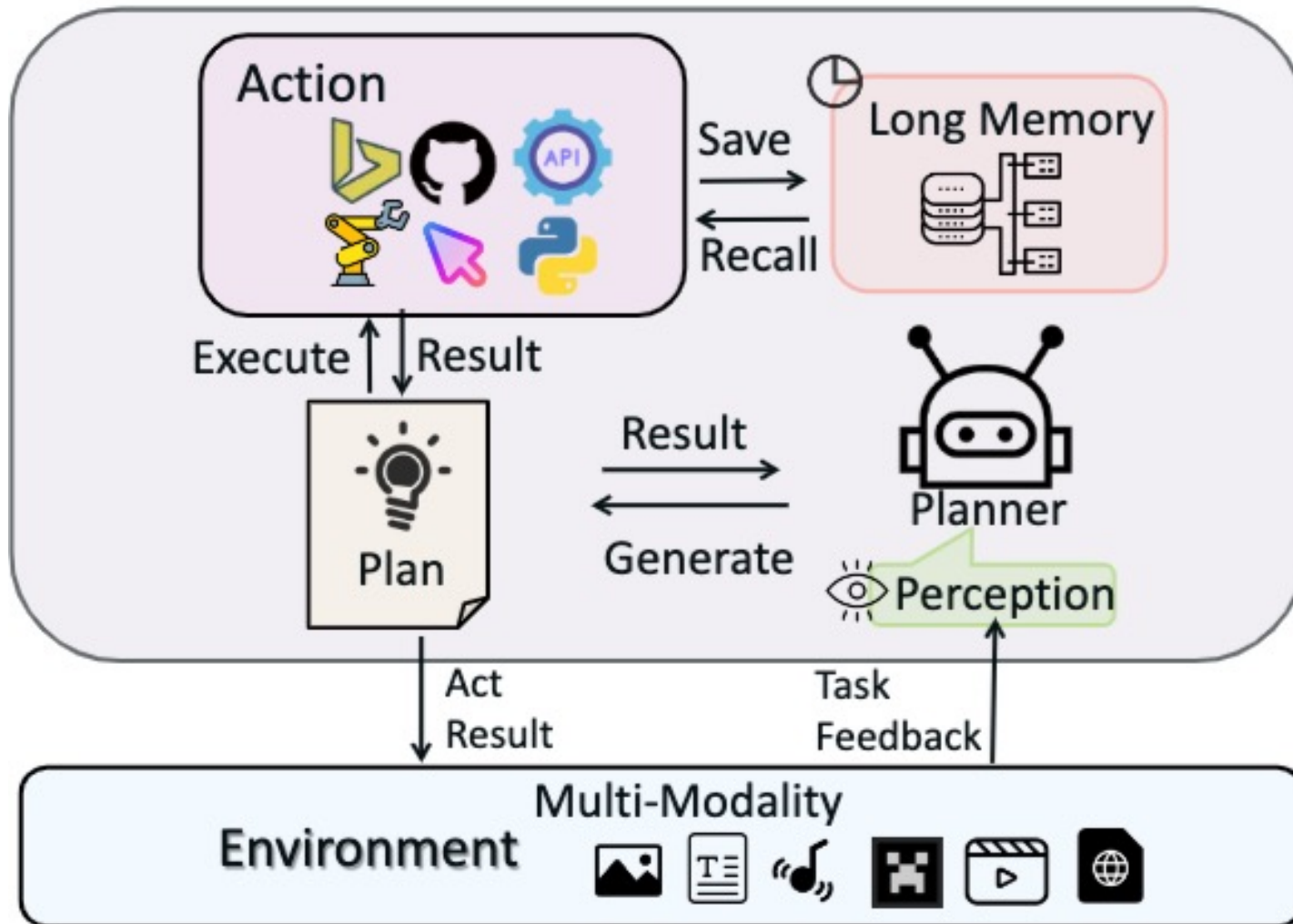
(b) Type II: **Finetuned LLMs** as Planners w/o long-term Memory.



Use action-related data to finetune existing open-source large models, enabling them to achieve decision-making, planning, and tool invocation capabilities comparable to closed-source LLMs

Large Multimodal Agents (LMAs)

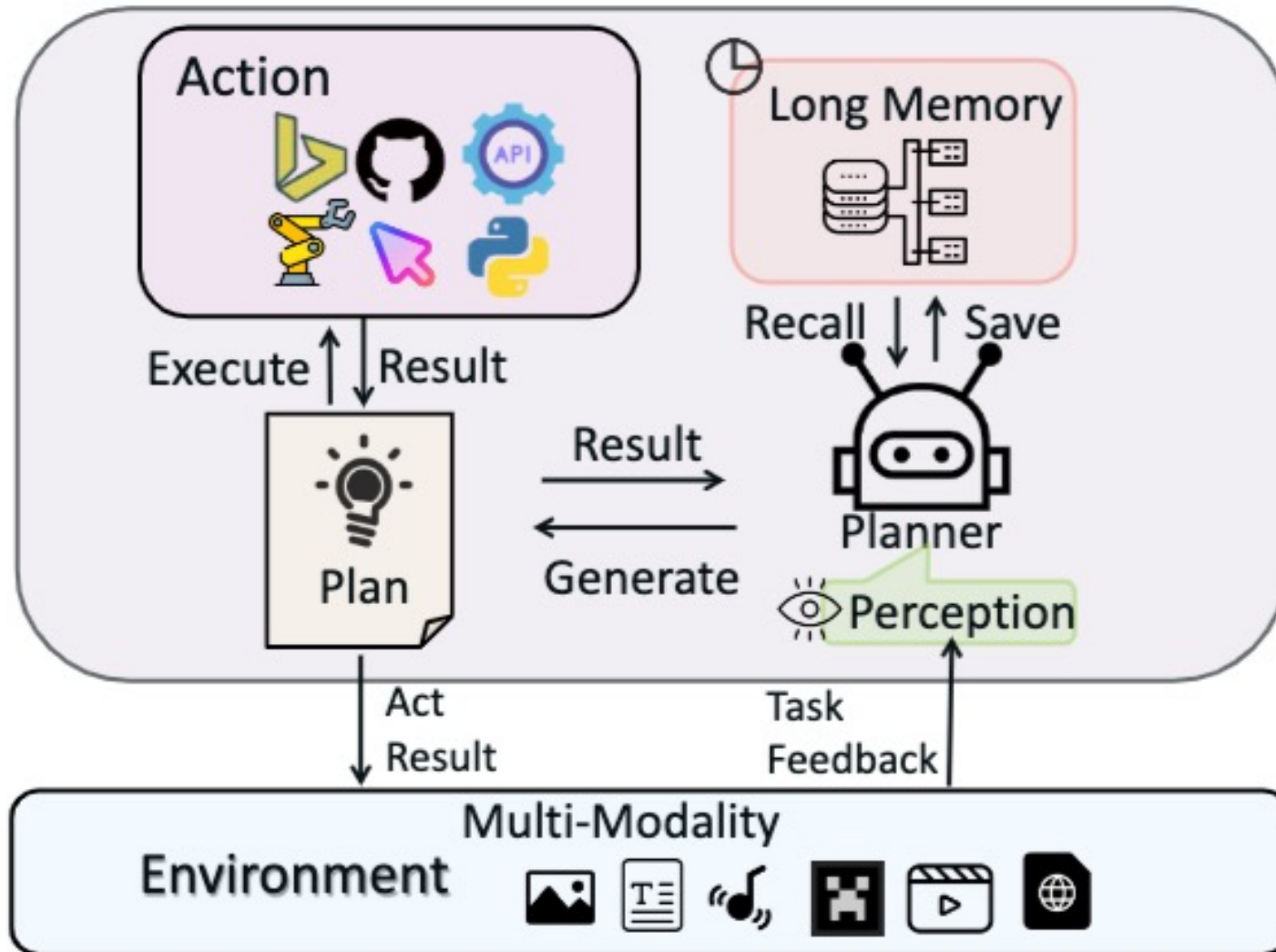
(c) Type III: Planners with **Indirect Long-term Memory**



Introduce indirect long-term memory functions, further enhancing their generalization and adaptation abilities in environments closer to the real world.

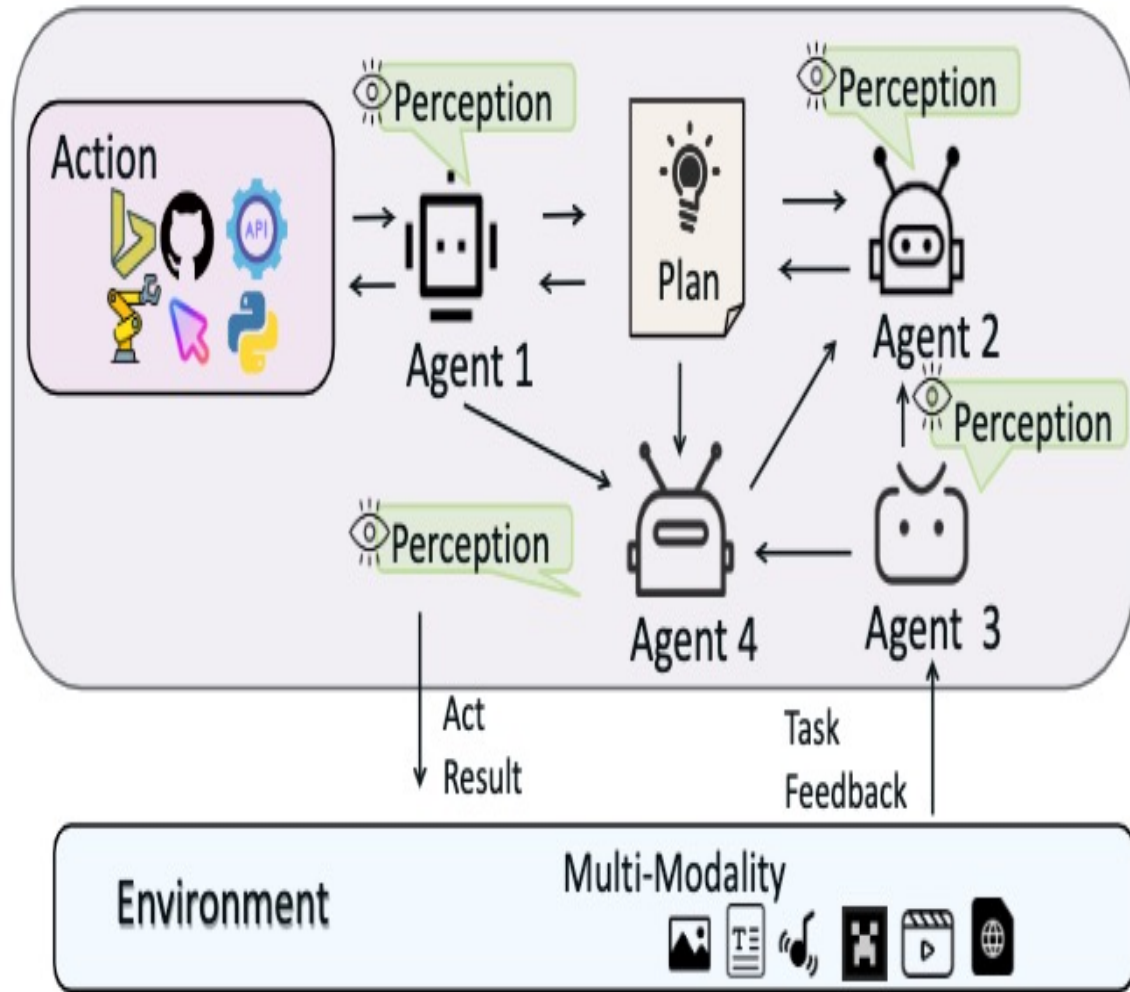
Large Multimodal Agents (LMAs)

(d) Type IV: Planners with **Native Long-term Memory**

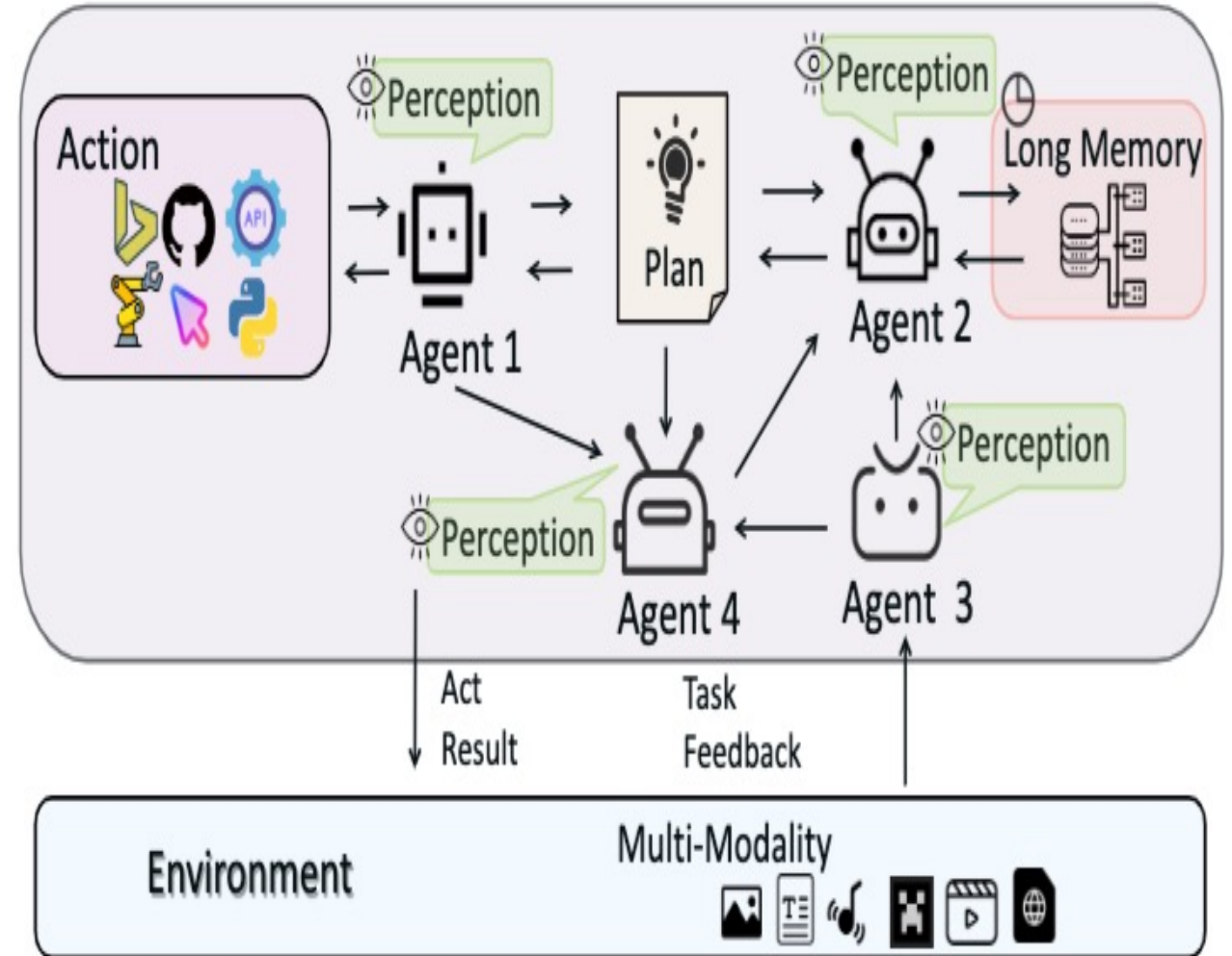


Introduce native long-term memory functions, further enhancing their generalization and adaptation abilities in environments closer to the real world.

Multi-Agent Frameworks

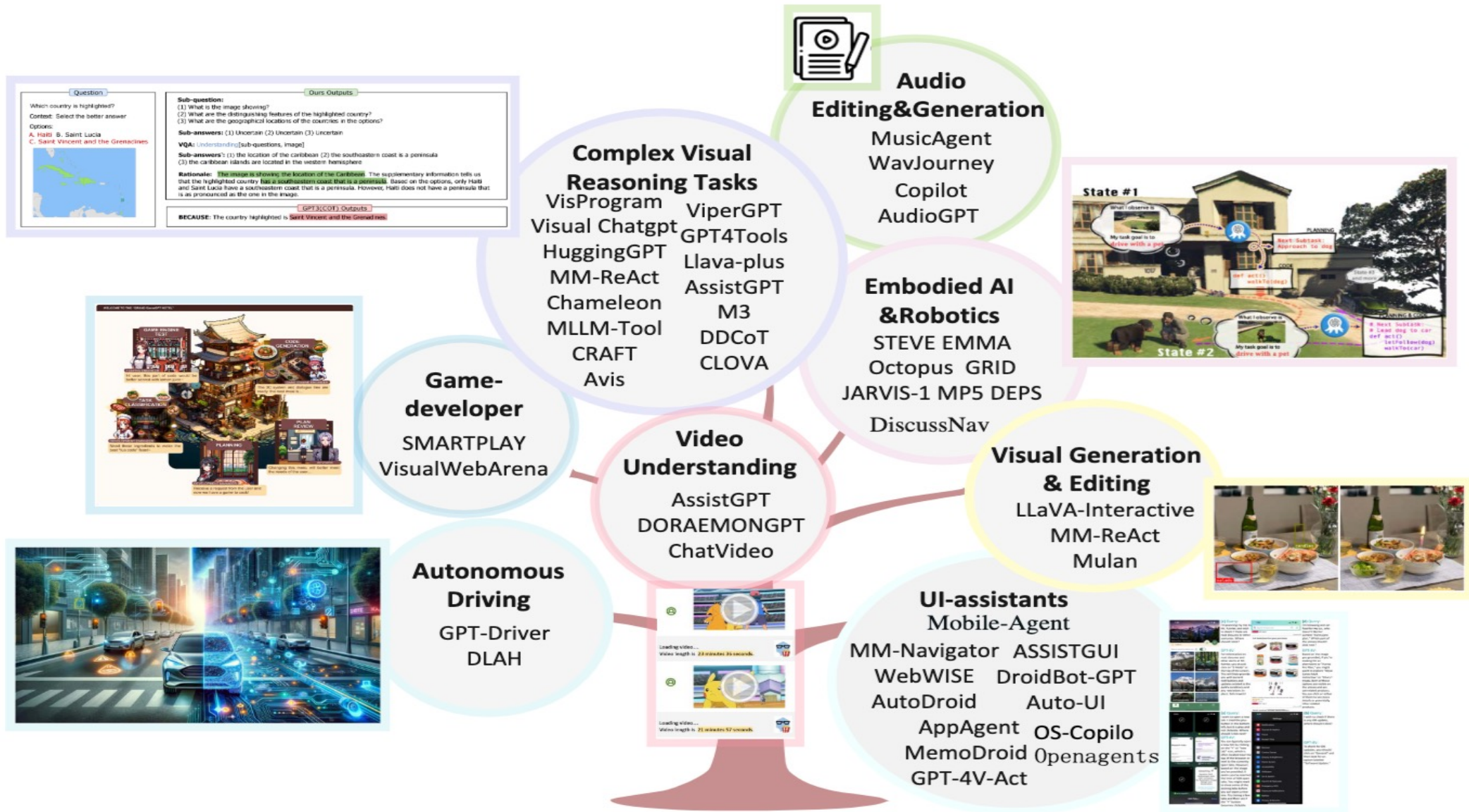


(a)

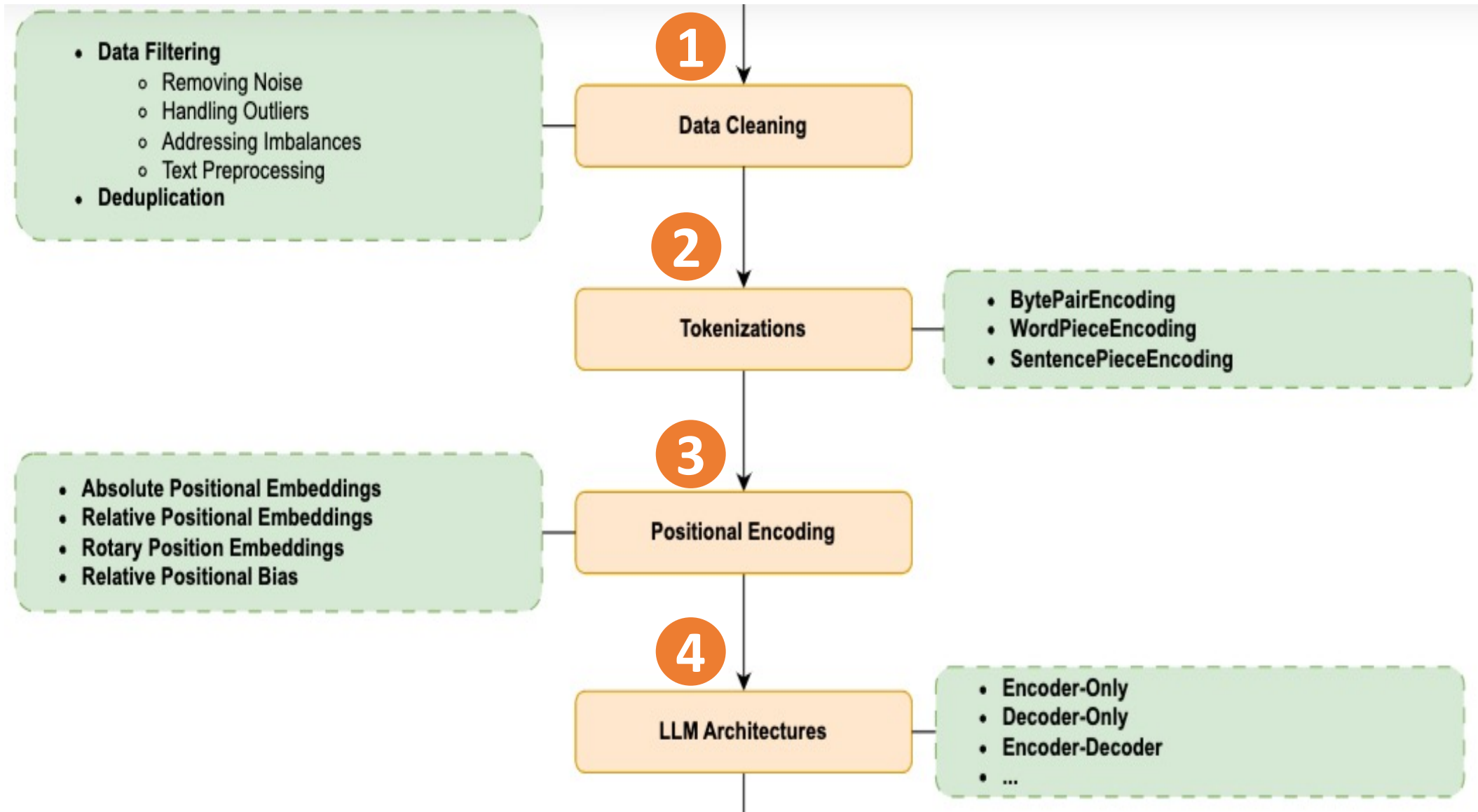


(b)

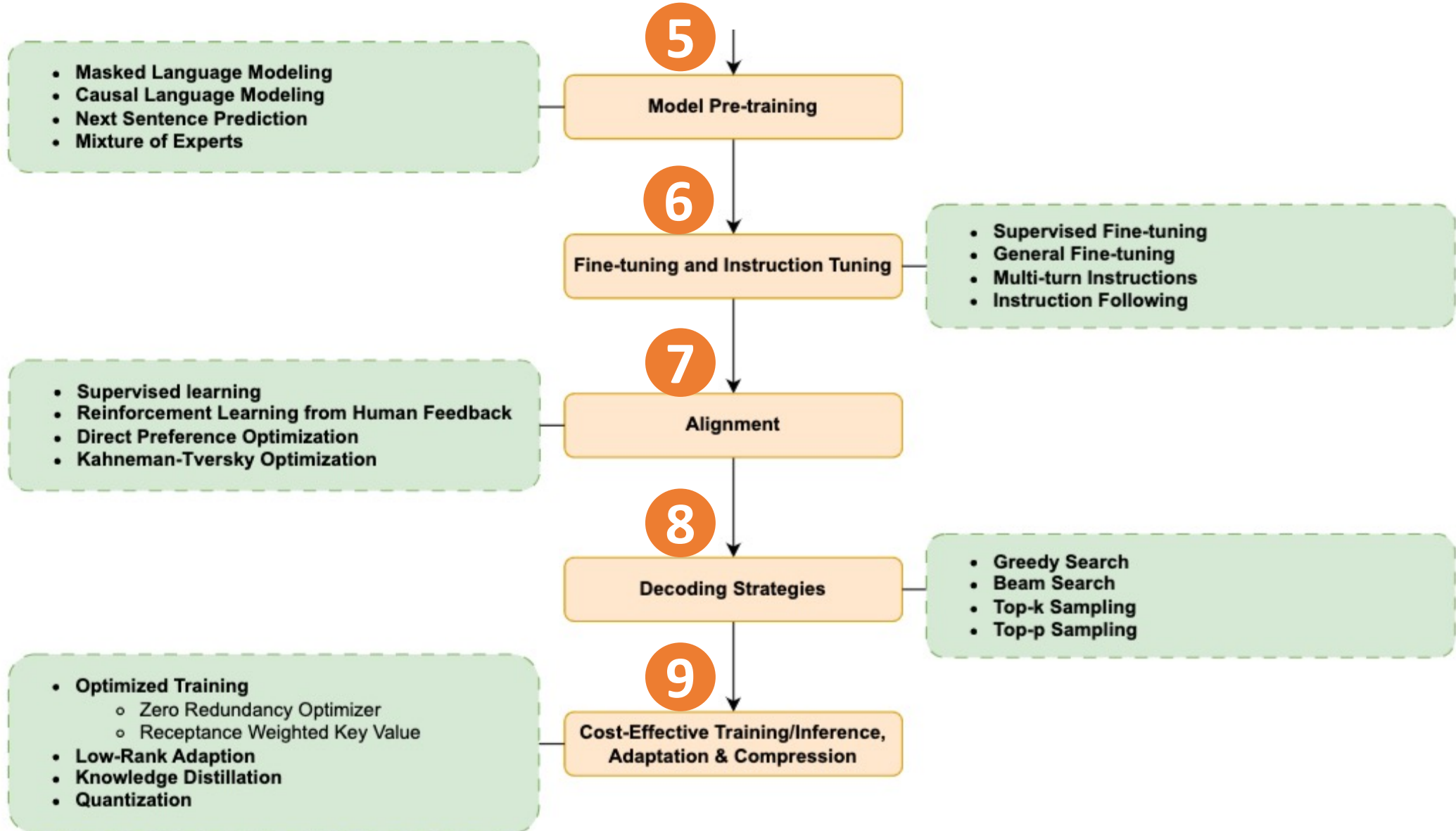
Applications of Large Multimodal Agents (LMAs)



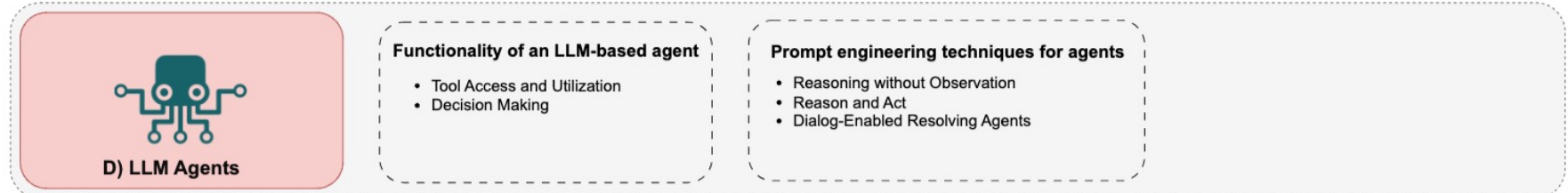
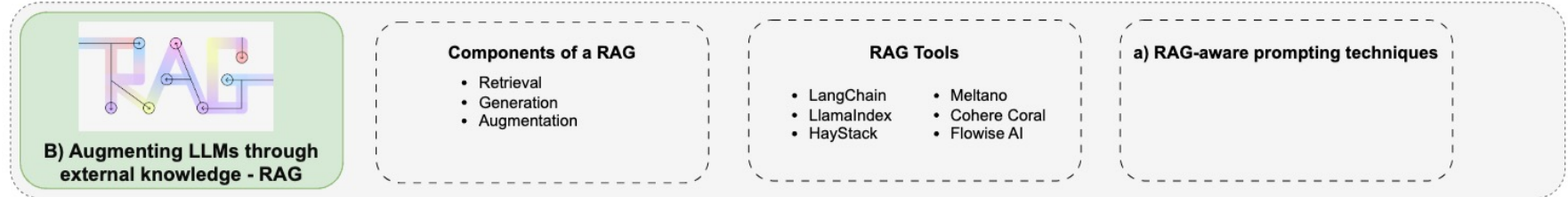
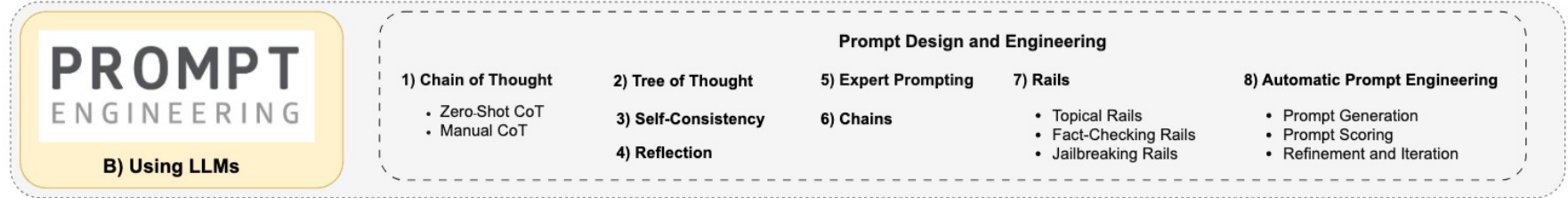
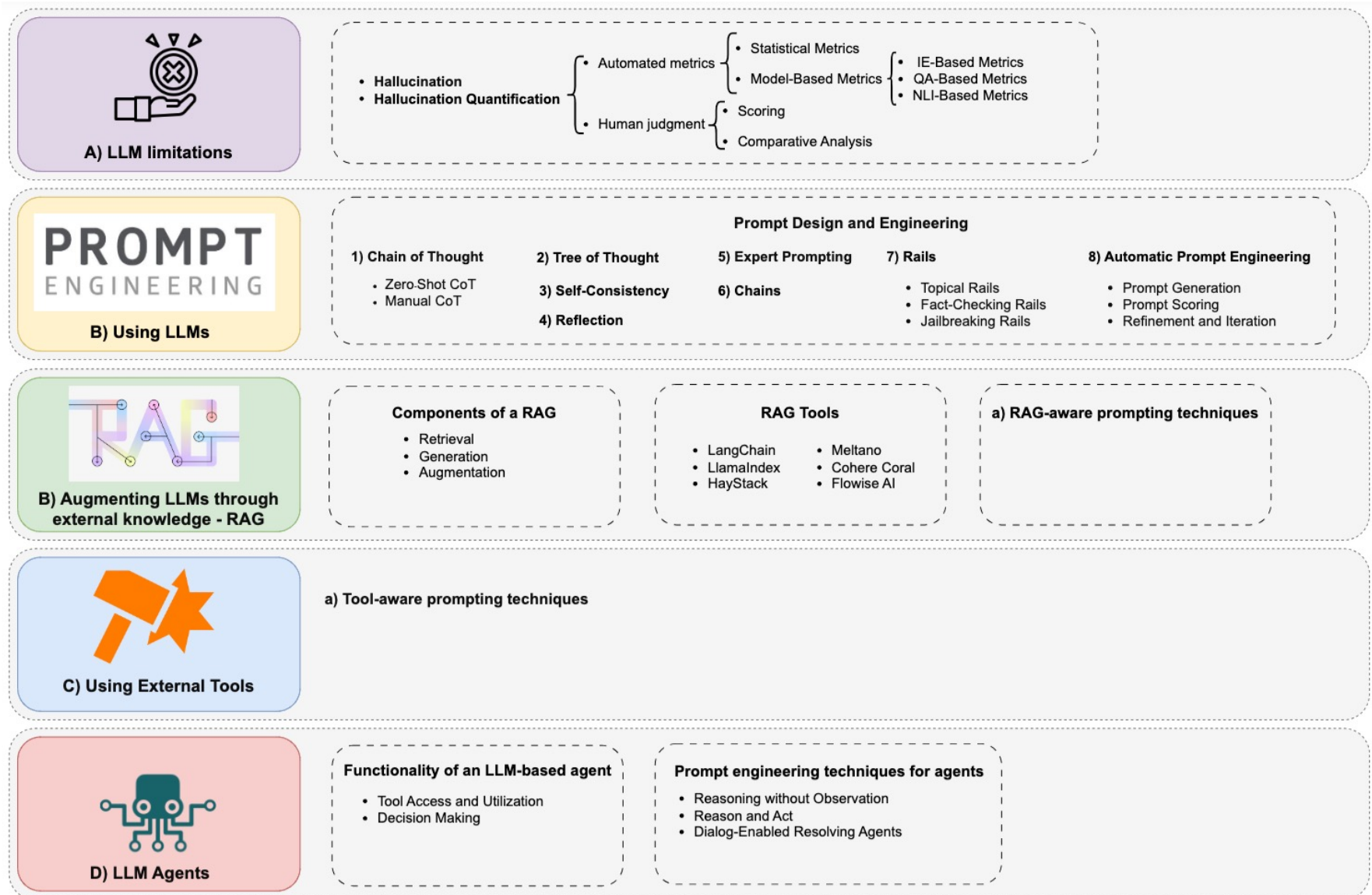
How LLMs Are Built?



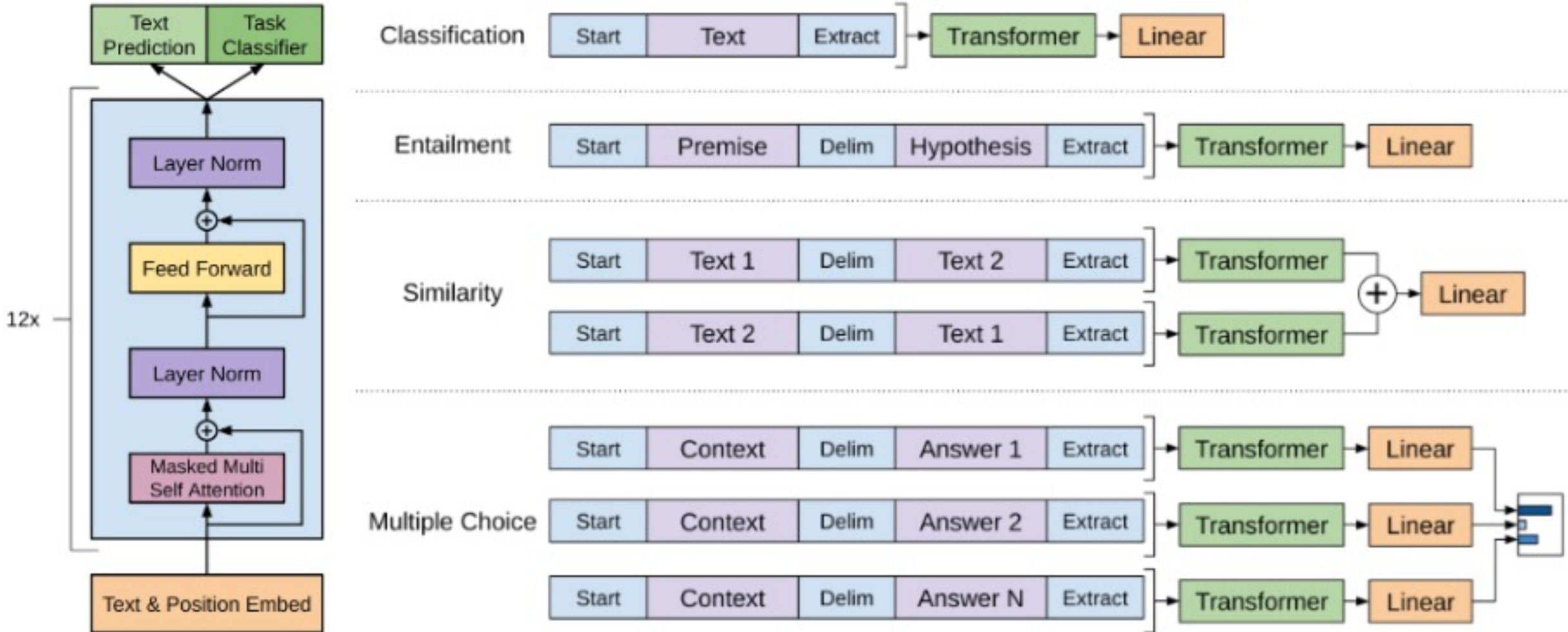
How LLMs Are Built?



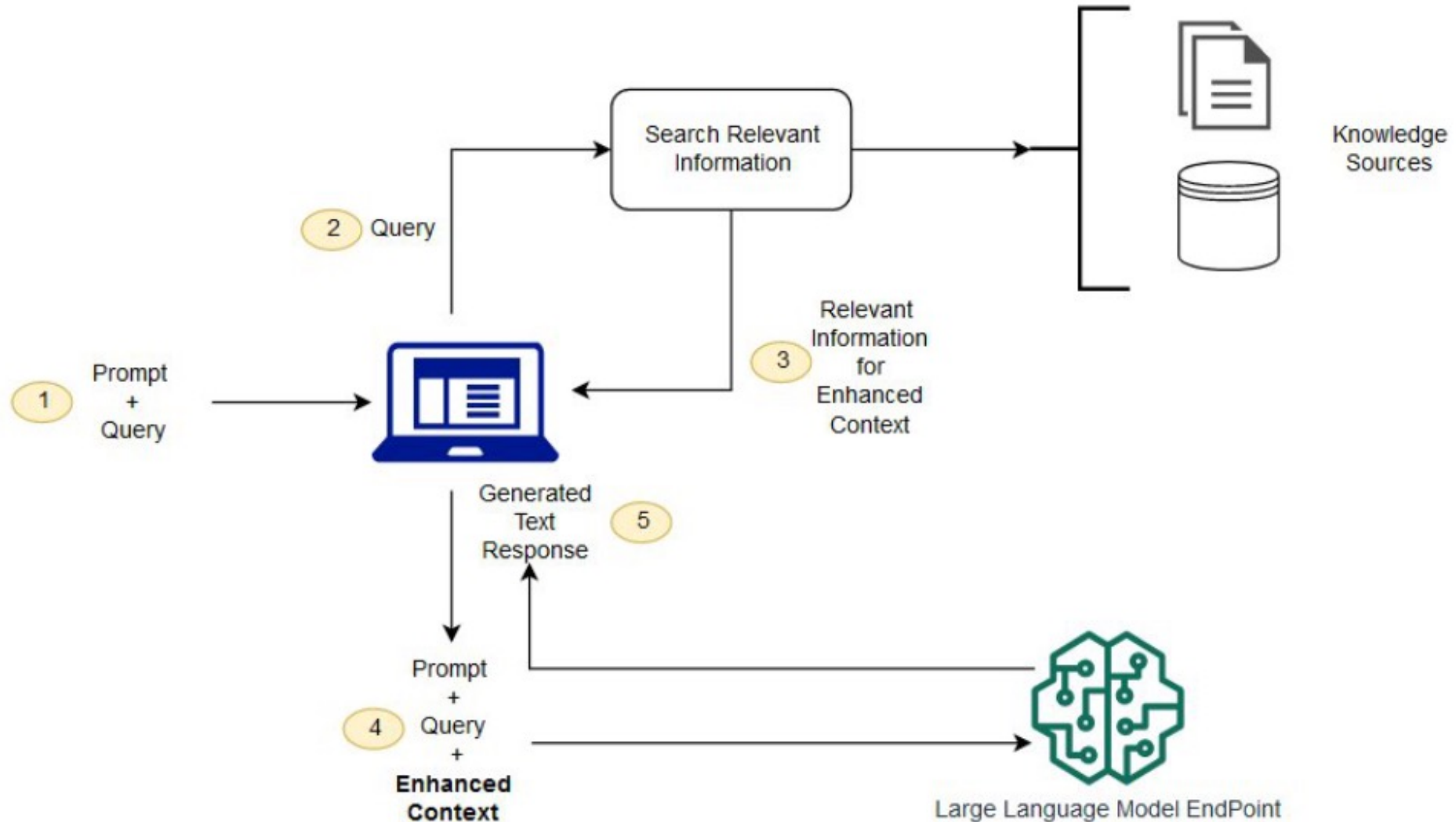
How LLMs Are Used and Augmented



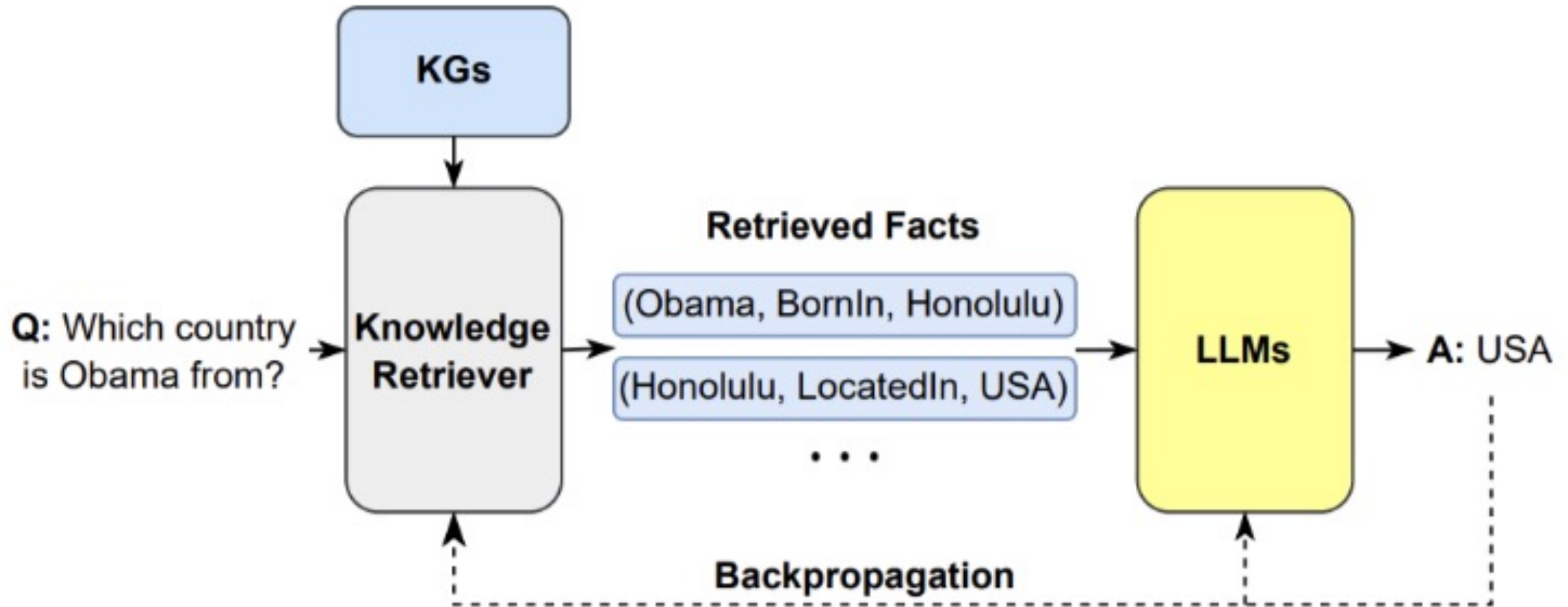
GPT Pretraining, and Fine-tuning Steps



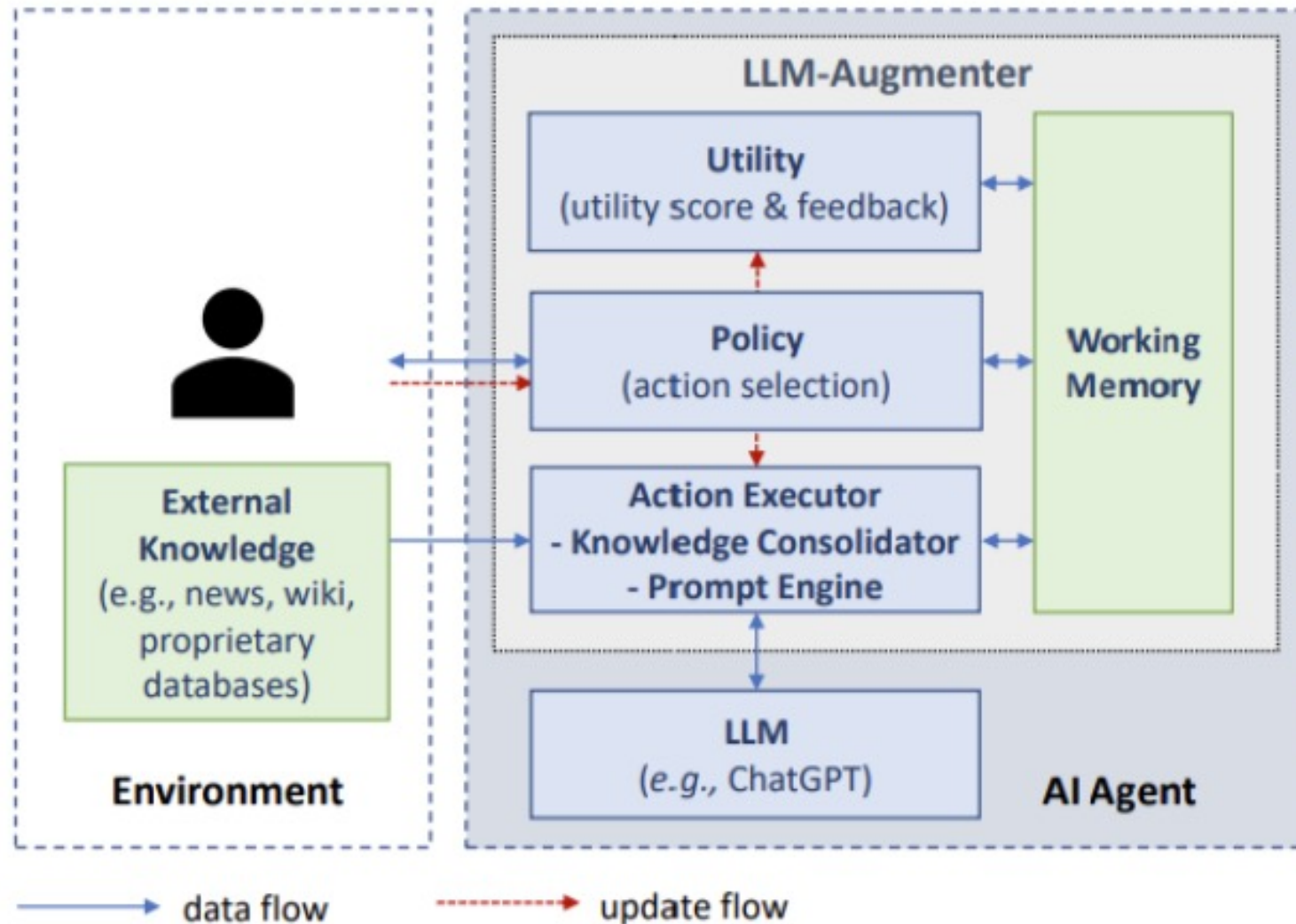
Synthesizing RAG with LLMs for Question Answering Application



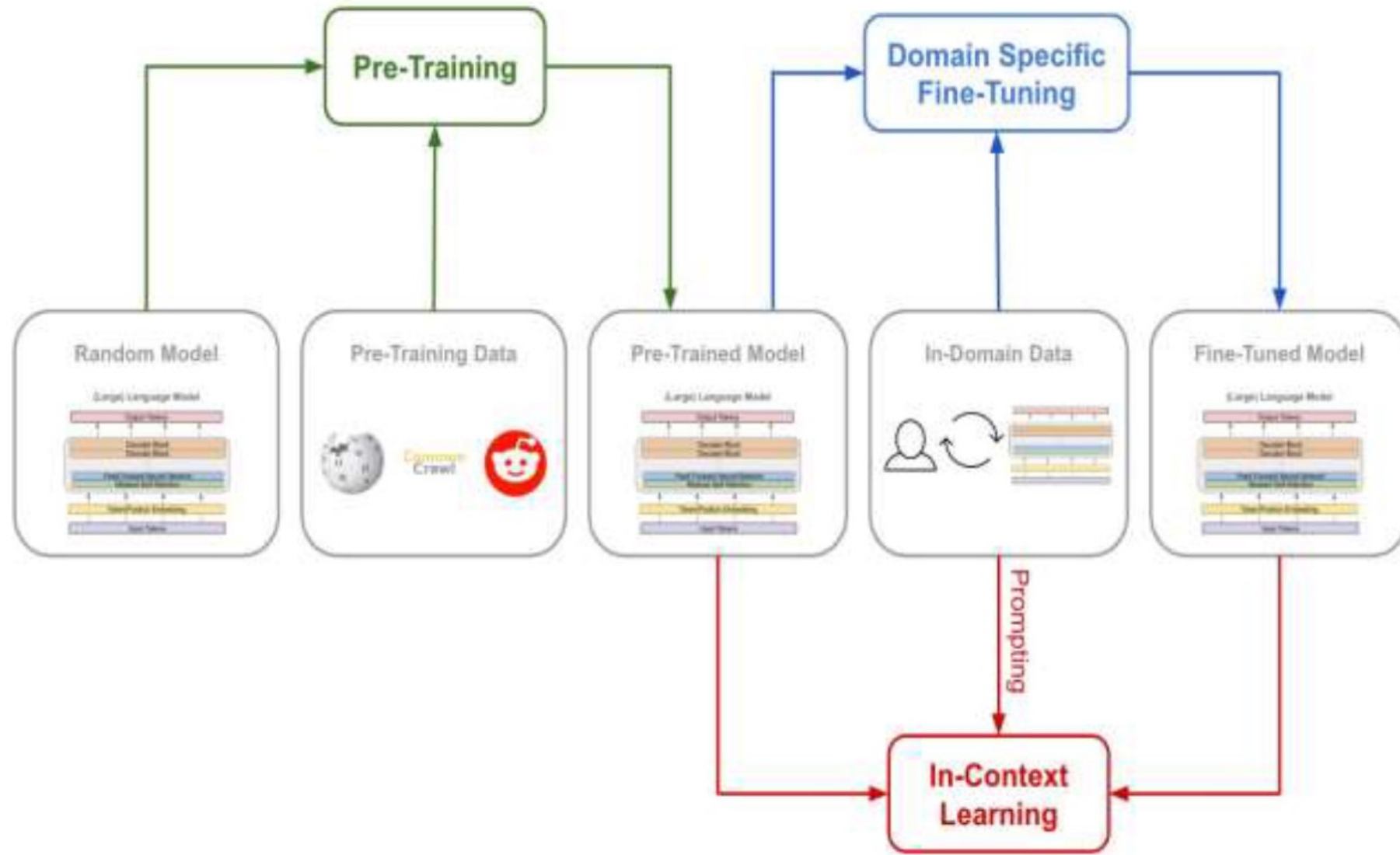
Synthesizing the KG as a Retriever with LLMs



LLM-based Agent for Conversational Information Seeking

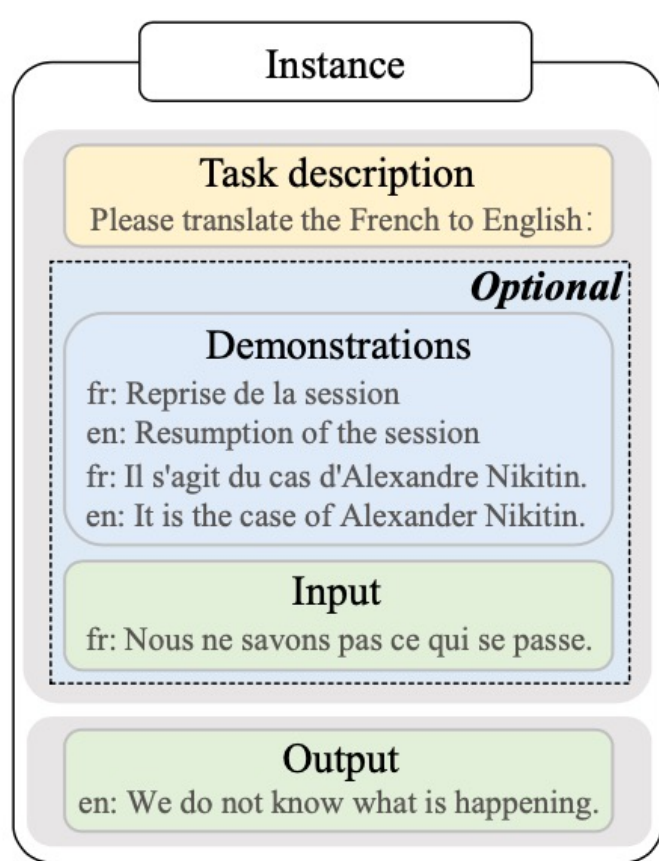


Data Gathering and Pre-training for LLMs

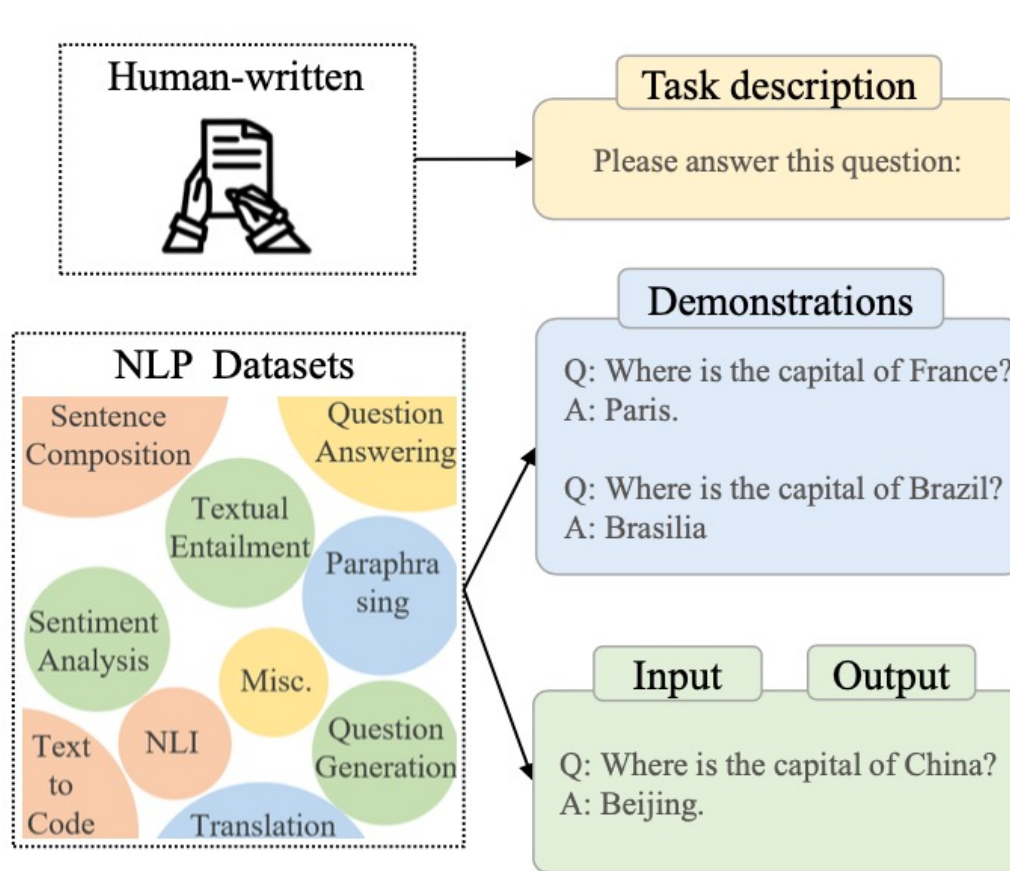


Pre-train, Prompt, and Predict: Prompting Methods in Natural Language Processing (LLMs)

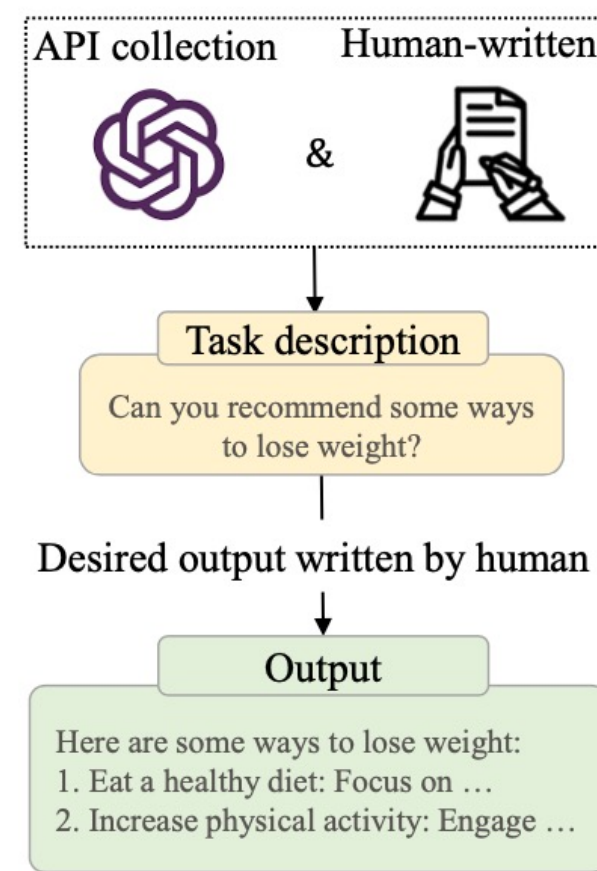
Instance Formatting and Two Different Methods for Constructing the Instruction-formatted Instances



(a) Instance format



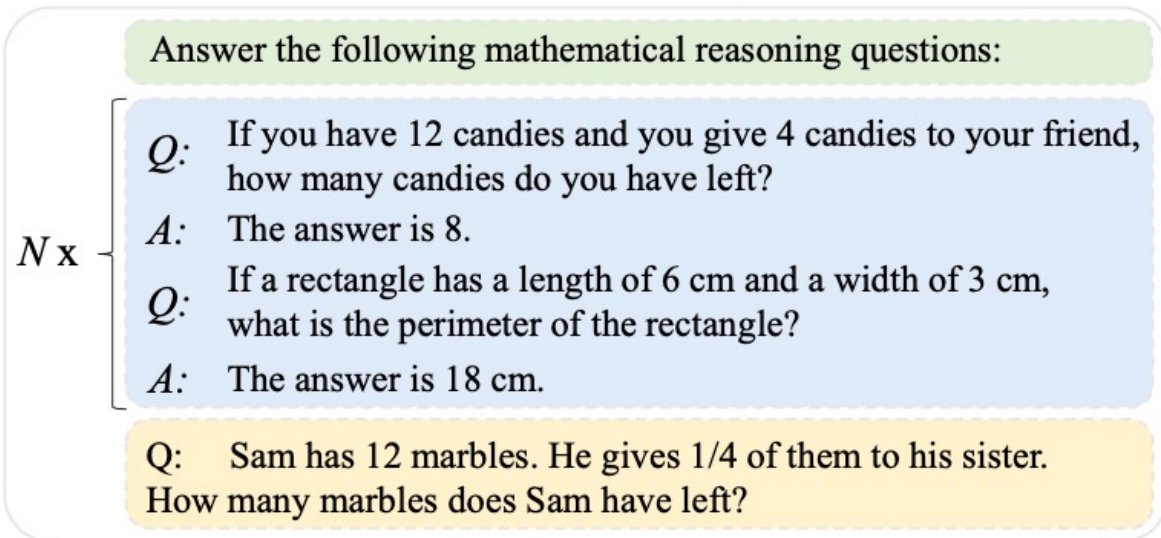
(b) Formatting existing datasets



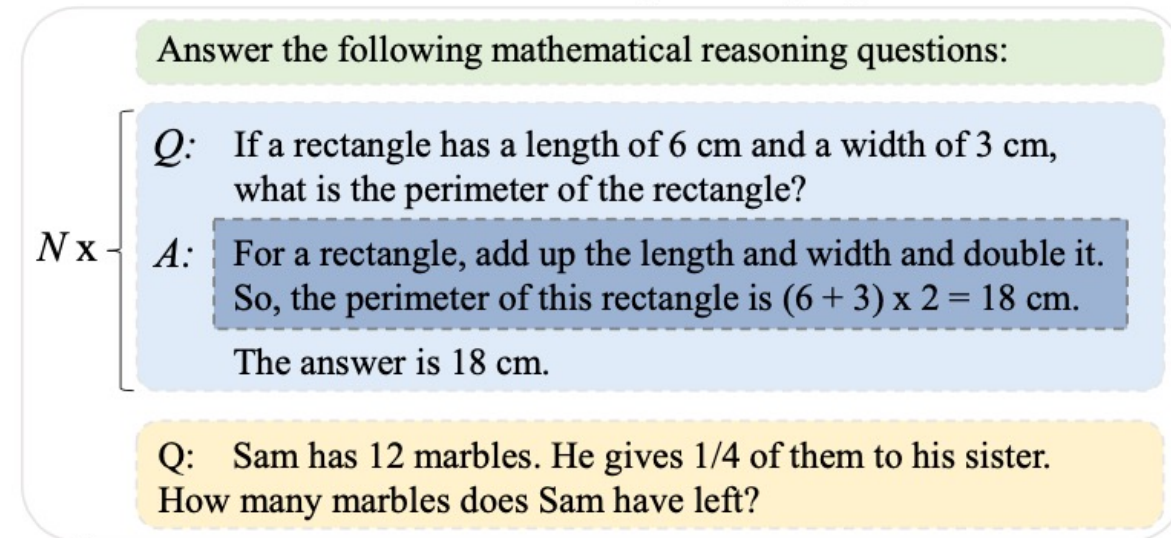
(c) Formatting human needs

In-context Learning (ICL) and Chain-of-thought (CoT) Prompting

In-Context Learning



Chain-of-Thought Prompting

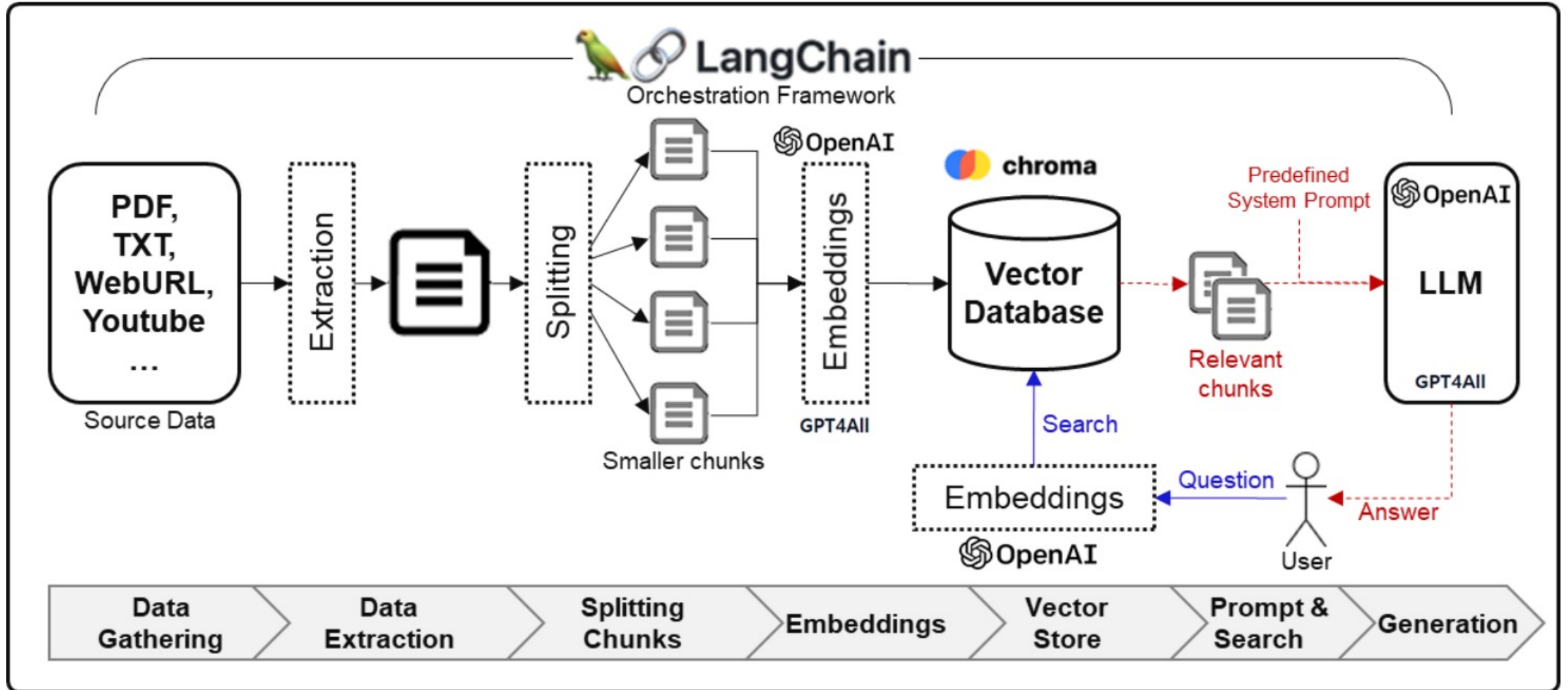


 : Task description  : Demonstration  : Chain-of-Thought  : Query

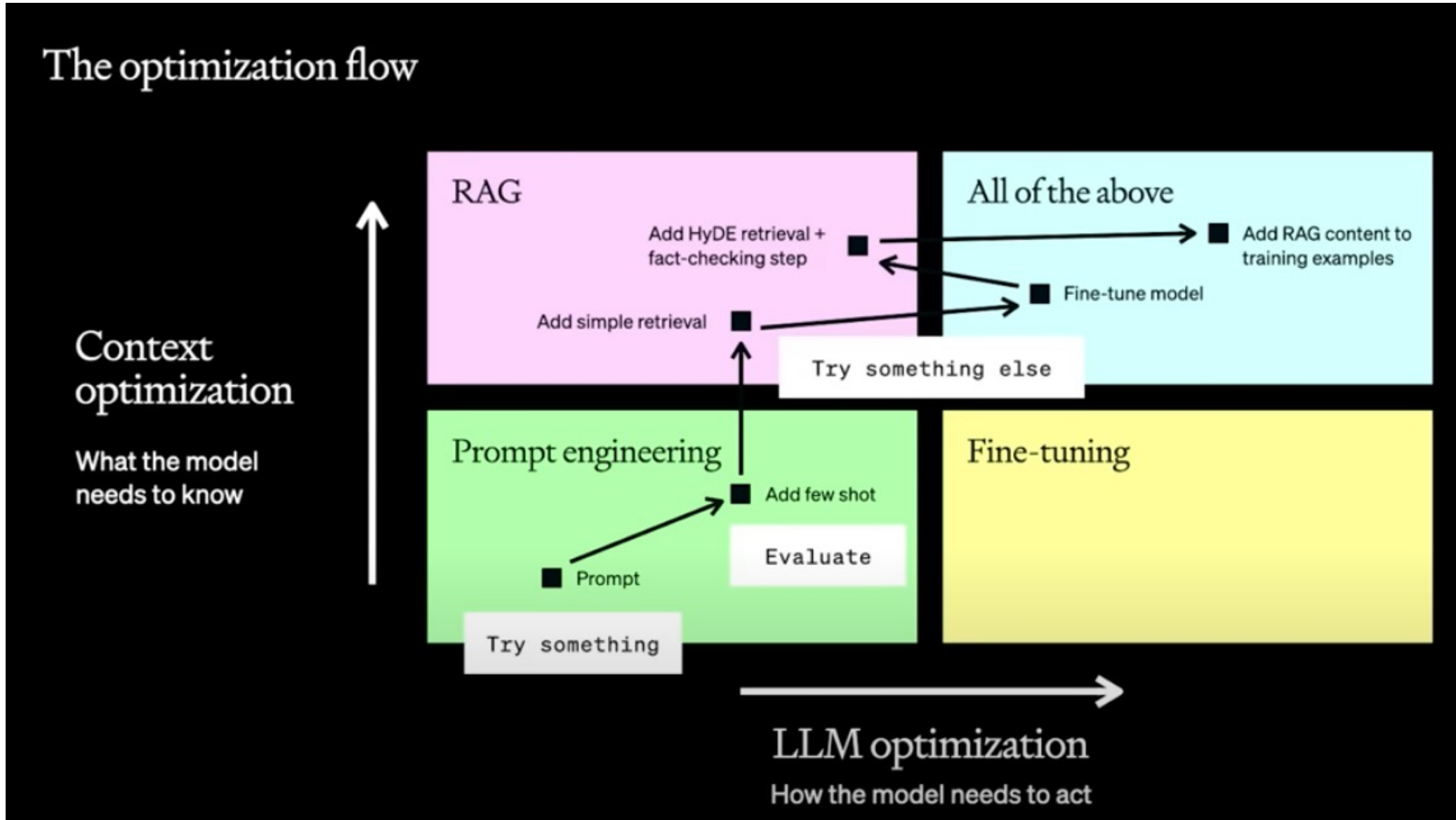
Four Paradigms in NLP (LM)

Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Feature (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
Transfer Learning: Pre-training, Fine-Tuning (FT)		
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
GAI: Pre-train, Prompt, and Predict (Prompting)		
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

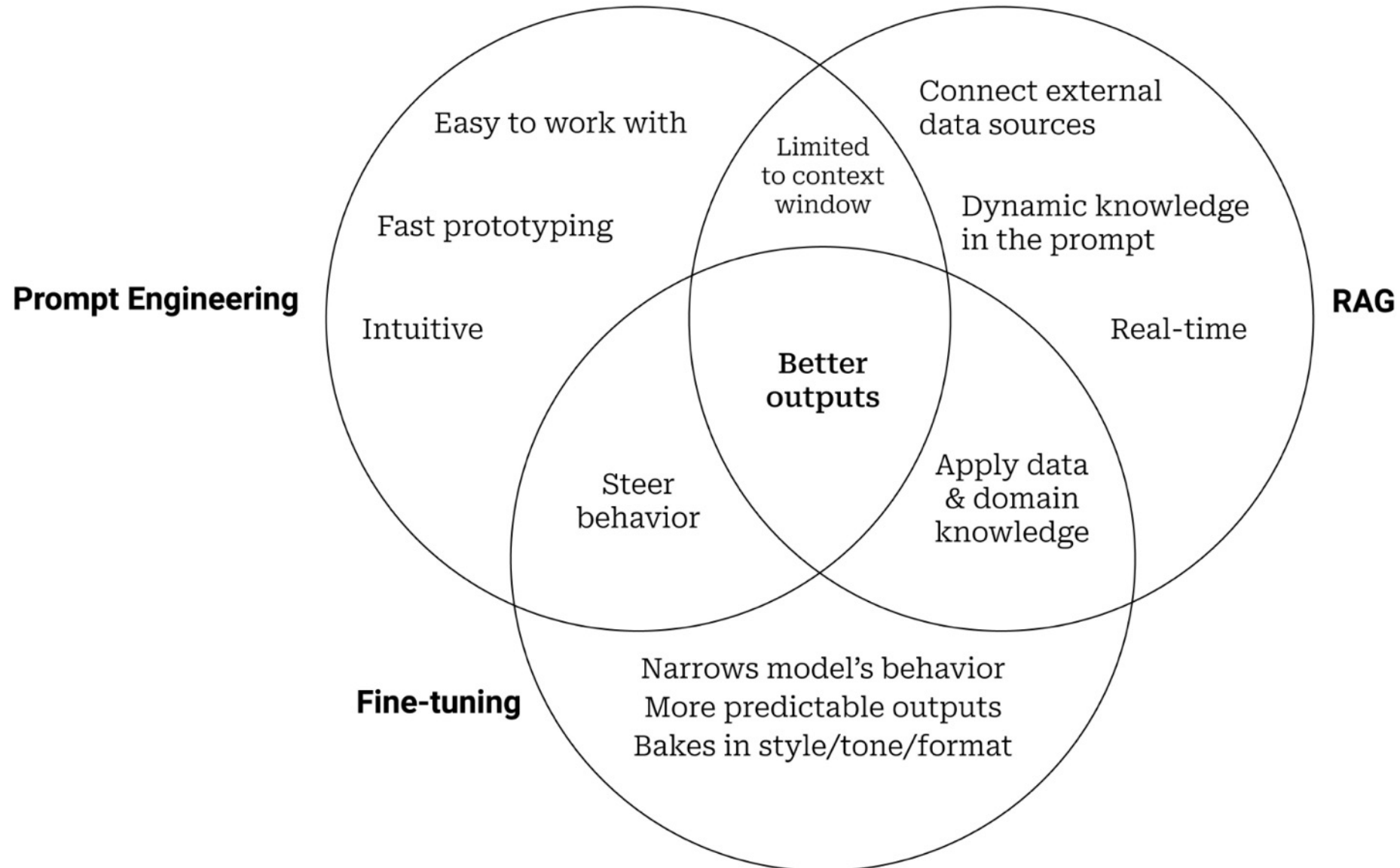
Framework for Implementing Generative AI Services using RAG Model



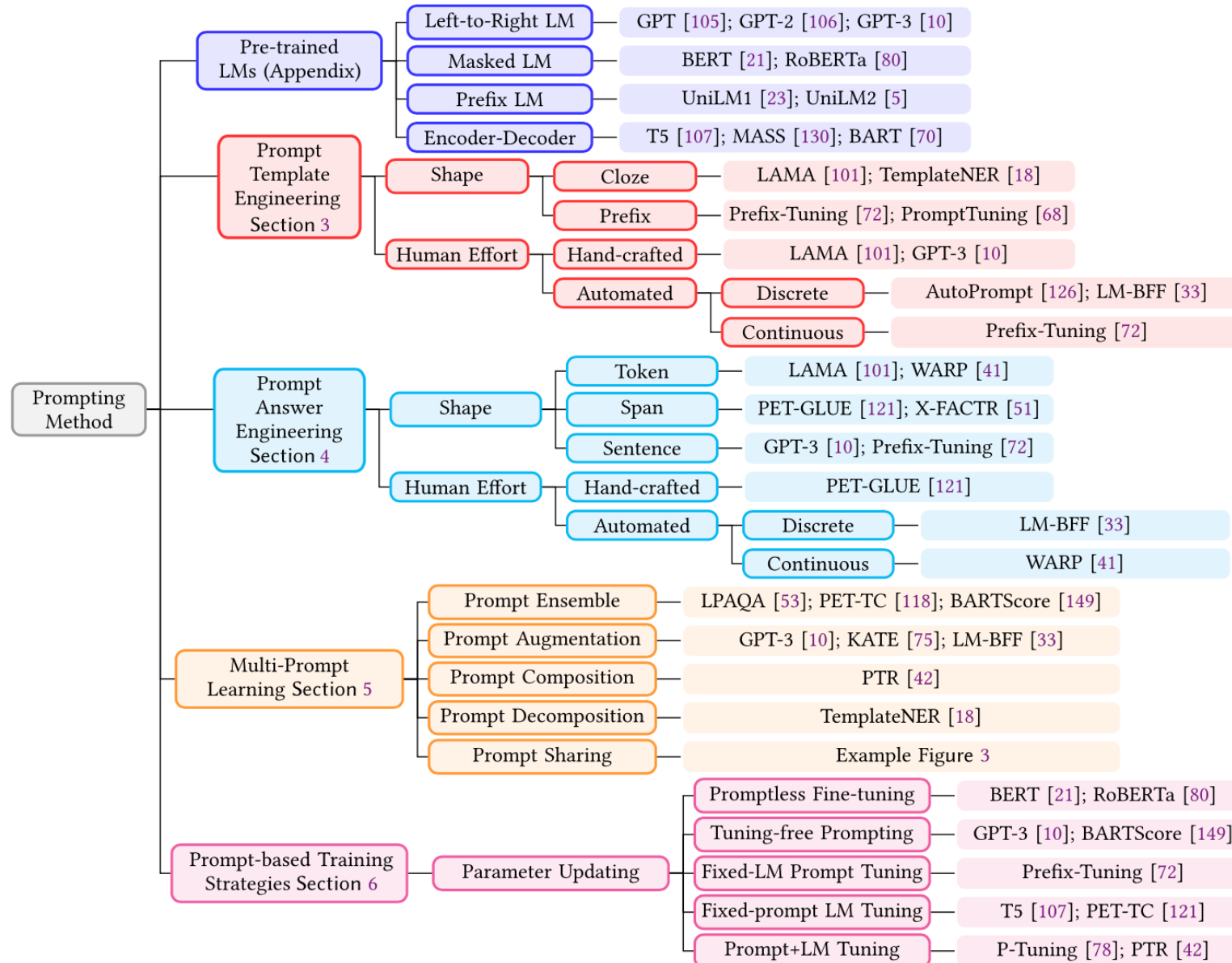
Maximizing LLM Performance



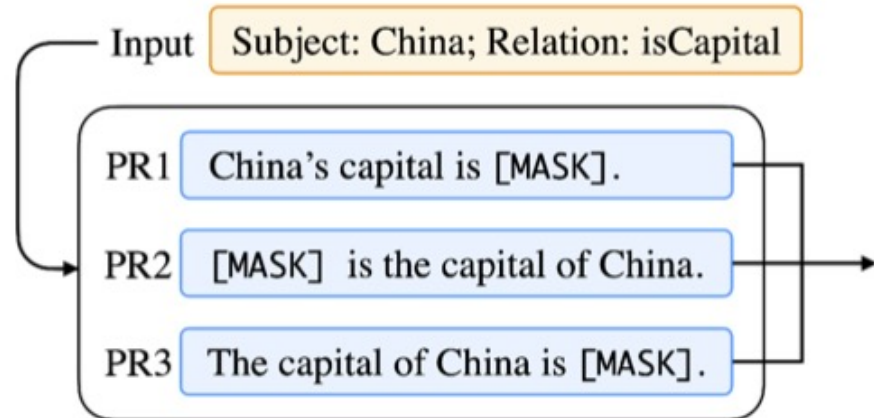
Prompt Engineering, Fine-tuning, and RAG



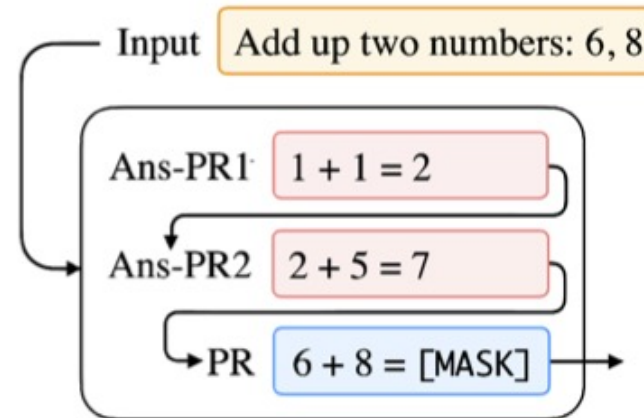
Typology of Prompting Methods



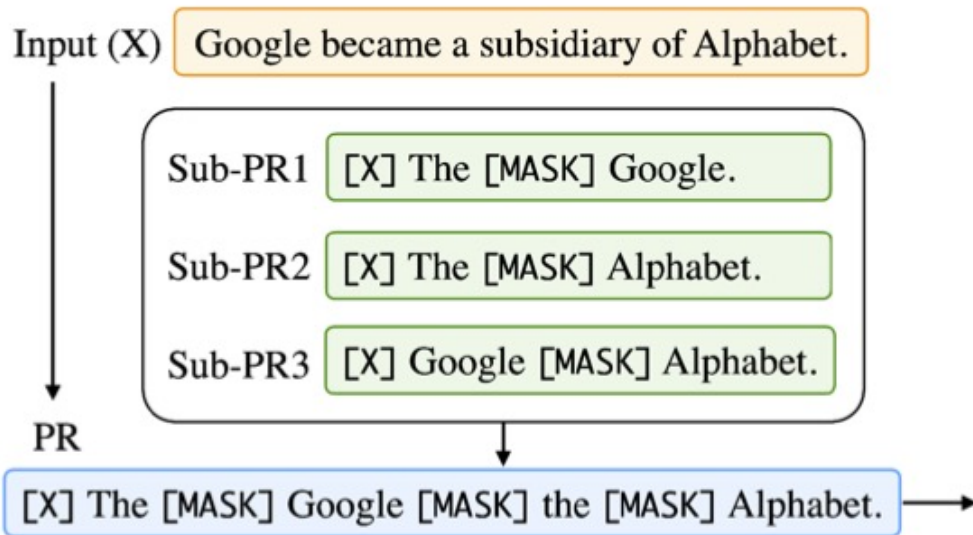
Different Multi-Prompt Learning Strategies



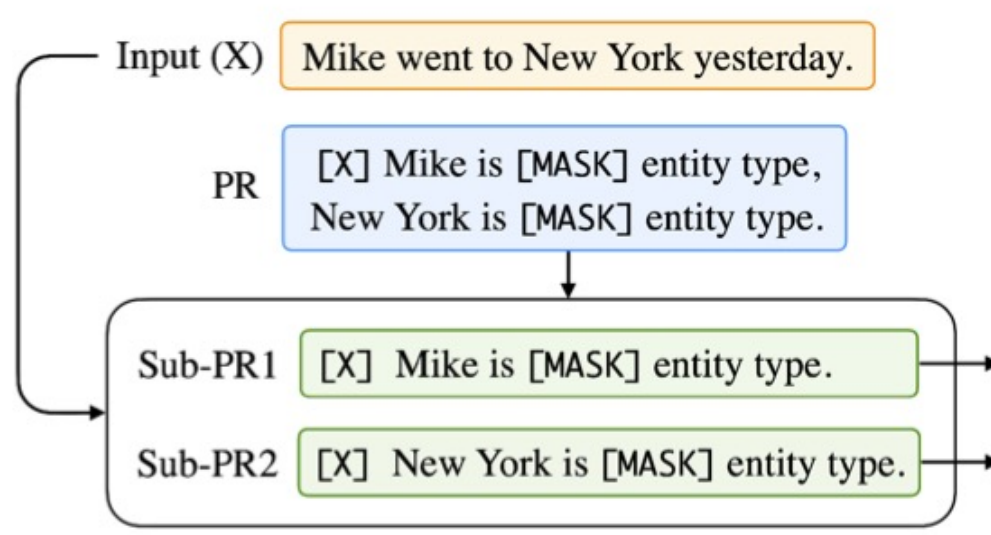
(a) Prompt Ensembling.



(b) Prompt Augmentation.



(c) Prompt Composition.



(d) Prompt Decomposition.

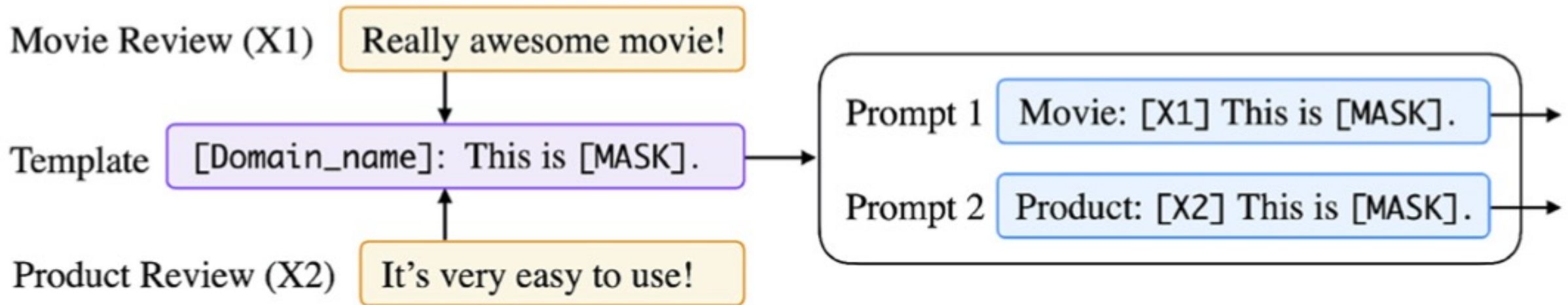
Examples of Input, Template, and Answer for Different Tasks

Type	Task Example	Input ([X])	Template	Answer ([Z])
Text Classification	Sentiment	I love this movie.	[X] The movie is [Z].	great fantastic ...
	Topics	He prompted the LM.	[X] The text is about [Z].	sports science ...
	Intention	What is taxi fare to Denver?	[X] The question is about [Z].	quantity city ...
Text-span Classification	Aspect Sentiment	Poor service but good food.	[X] What about service? [Z].	Bad Terrible ...
Text-pair Classification	Natural Language Inference	[X1]: An old man with ... [X2]: A man walks ...	[X1]? [Z], [X2]	Yes No ...
Tagging	Named Entity Recognition	[X1]: Mike went to Paris. [X2]: Paris	[X1][X2] is a [Z] entity.	organization location ...
Text Generation	Summarization	Las Vegas police ...	[X] TL;DR: [Z]	The victim ... A woman
	Translation	Je vous aime.	French: [X] English: [Z]	I love you. I fancy you. ...
Regression	Textual Similarity	[X1]: A man is smoking. [X2]: A man is skating.	[X1] [Z], [X2]	Yes No ...

Characteristics of Different Tuning Strategies

Strategy	LM Params	Prompt Params		Example
		Additional	Tuned	
Promptless Fine-tuning	Tuned	—		ELMo [97], BERT [20], BART [69]
Tuning-free Prompting	Frozen	✗	✗	GPT-3 [9], AutoPrompt [125], LAMA [100]
Fixed-LM Prompt Tuning	Frozen	✓	Tuned	Prefix-Tuning [71], Prompt-Tuning [67]
Fixed-prompt LM Tuning	Tuned	✗	✗	PET-TC [117], PET-Gen [118], LM-BFF [32]
Prompt+LM Fine-tuning	Tuned	✓	Tuned	PADA [5], P-Tuning [77], PTR [41]

Multi-prompt Learning for Multi-task, Multi-domain, or Multi-lingual Learning



GPT Prompt Engineering for Question Answering

- Find the answer to the question from the given context.
- When the question cannot be answered with the given context, say "unanswerable".
- Just say the answer without repeating the question.
- Context: {context}
- Question:{question}
- Answer:

Prompts and QA Inference For FLAN T5

Question Answering

- Prompts and QA Inference For FLAN T5, we follow [41] and use the following prompt:
- Context: {context}\nQuestion: {question}\nAnswer:
- Context: {context}
- Question: {question}
- Answer:

Prompt Engineering

- **Prompts For FLAN models**

- Longpre, S., Hou, L., Vu, T., Webson, A., Chung, H. W., Tay, Y., ... & Roberts, A. (2023). The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.

- **MNLI, NLI-FEVER, VitaminC:**

- "Premise: {premise}\n\nHypothesis: {hypothesis}\n\nDoes the premise entail the hypothesis?\n\nA yes\nB it is not possible to tell\nC no"

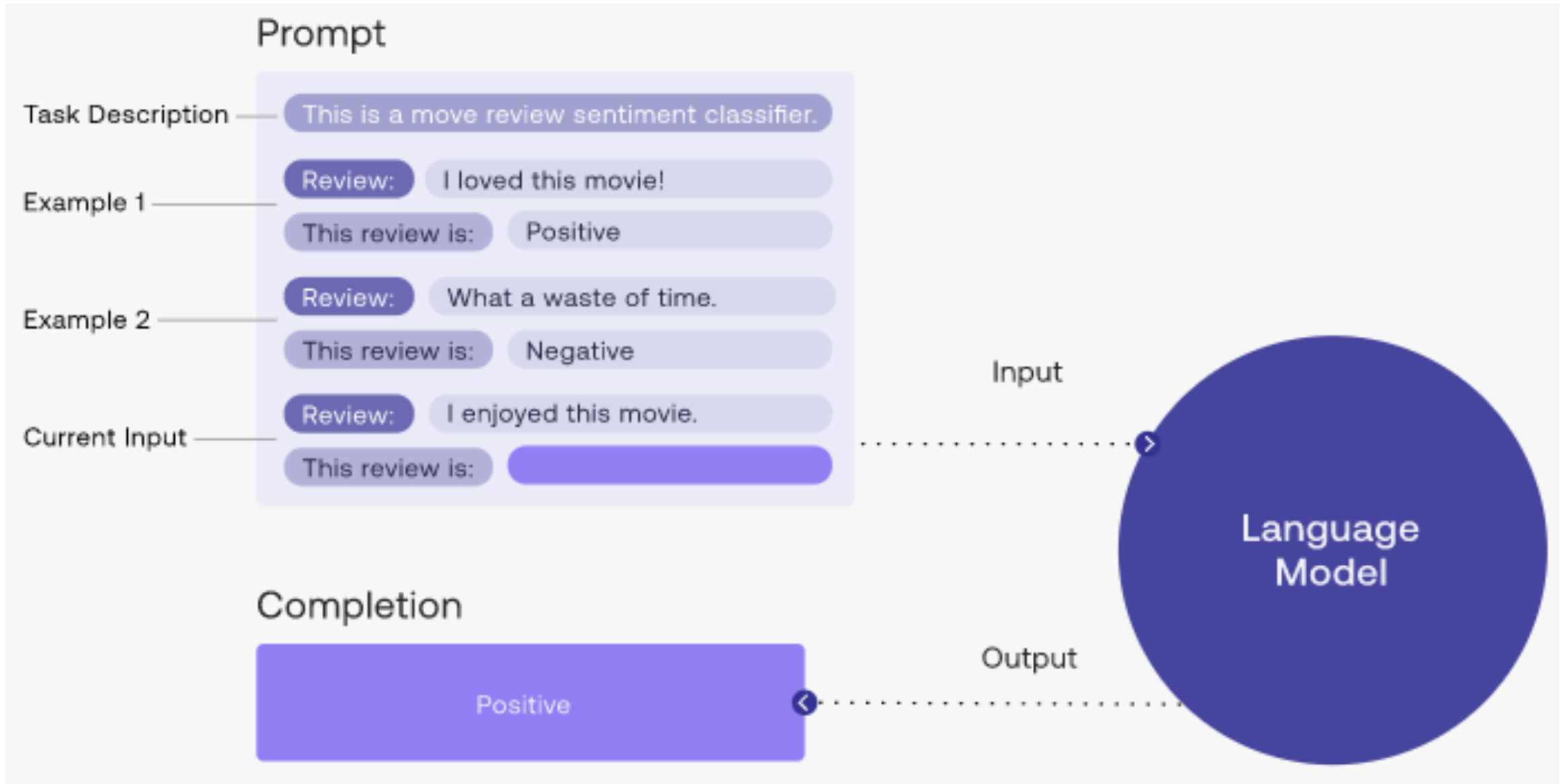
- **ANLI:**

- "{context}\n\nBased on the paragraph above can we conclude that \"{hypothesis}\"?\n\nA Yes\nB It's impossible to say\nC No"

- **SNLI:**

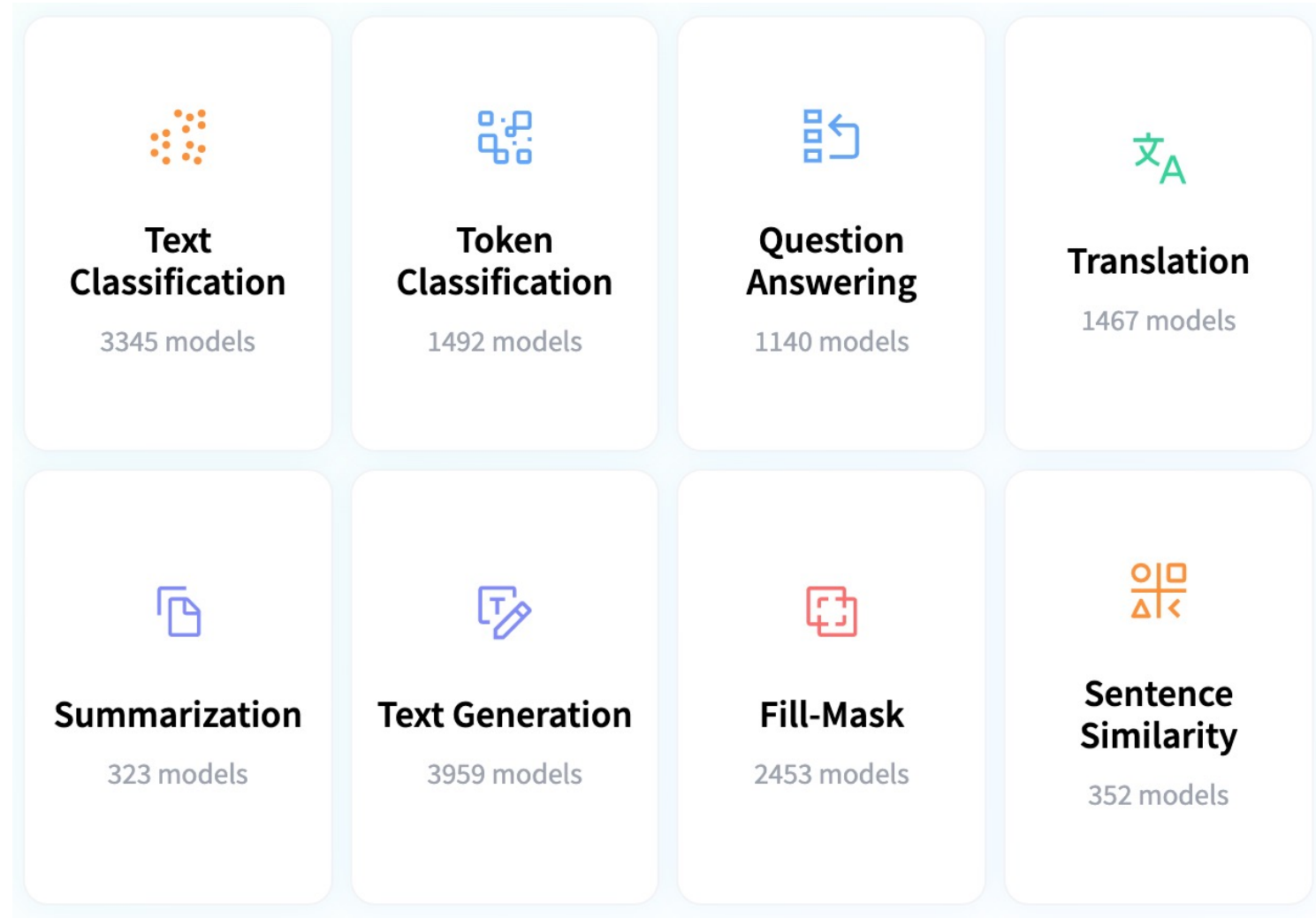
- "If \"{premise}\", does this mean that \"{hypothesis}\"?\n\nA yes\nB it is not possible to tell\nC no"

Prompt Engineering with ChatGPT for NLP



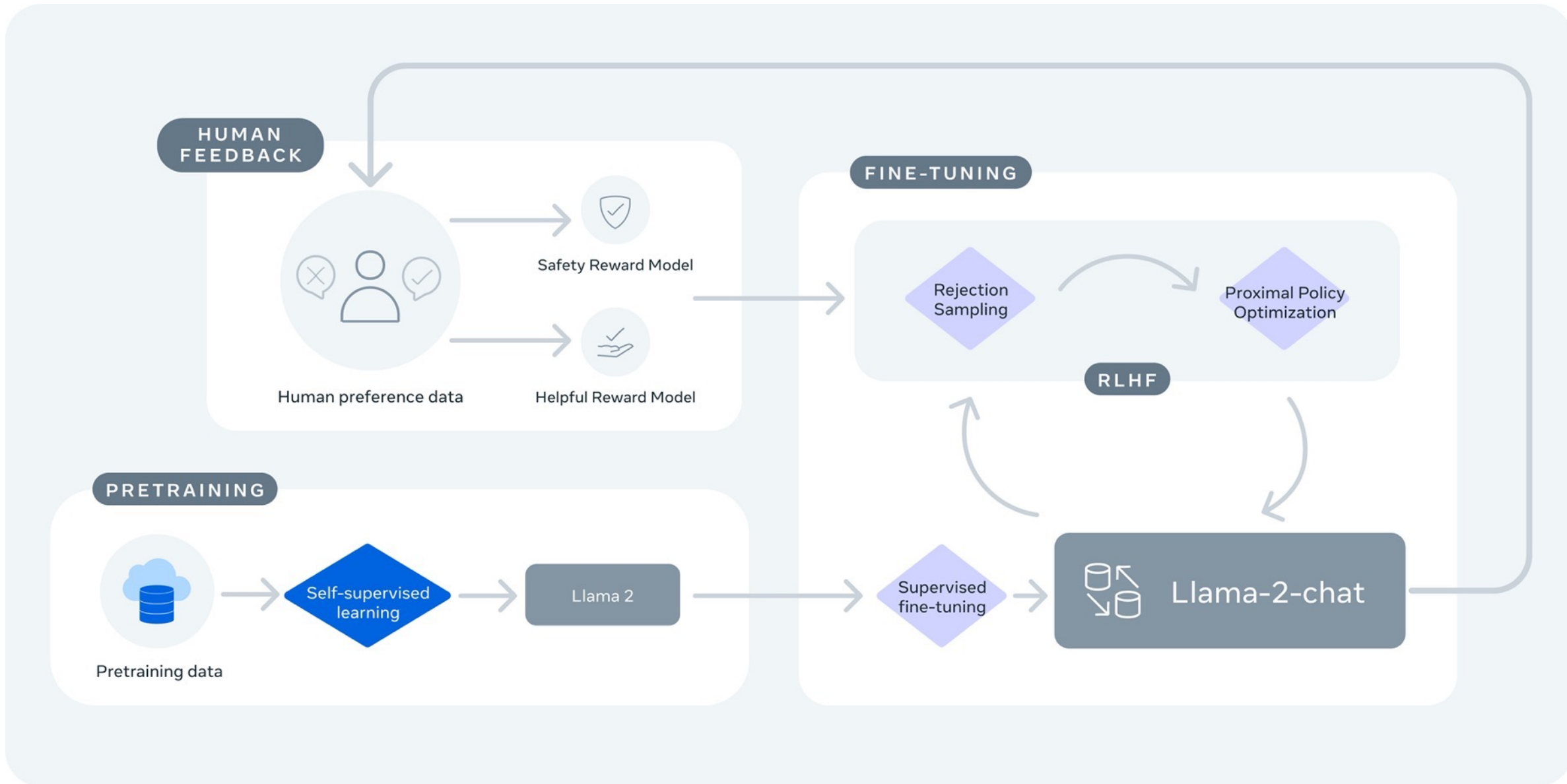
Hugging Face Tasks

Natural Language Processing

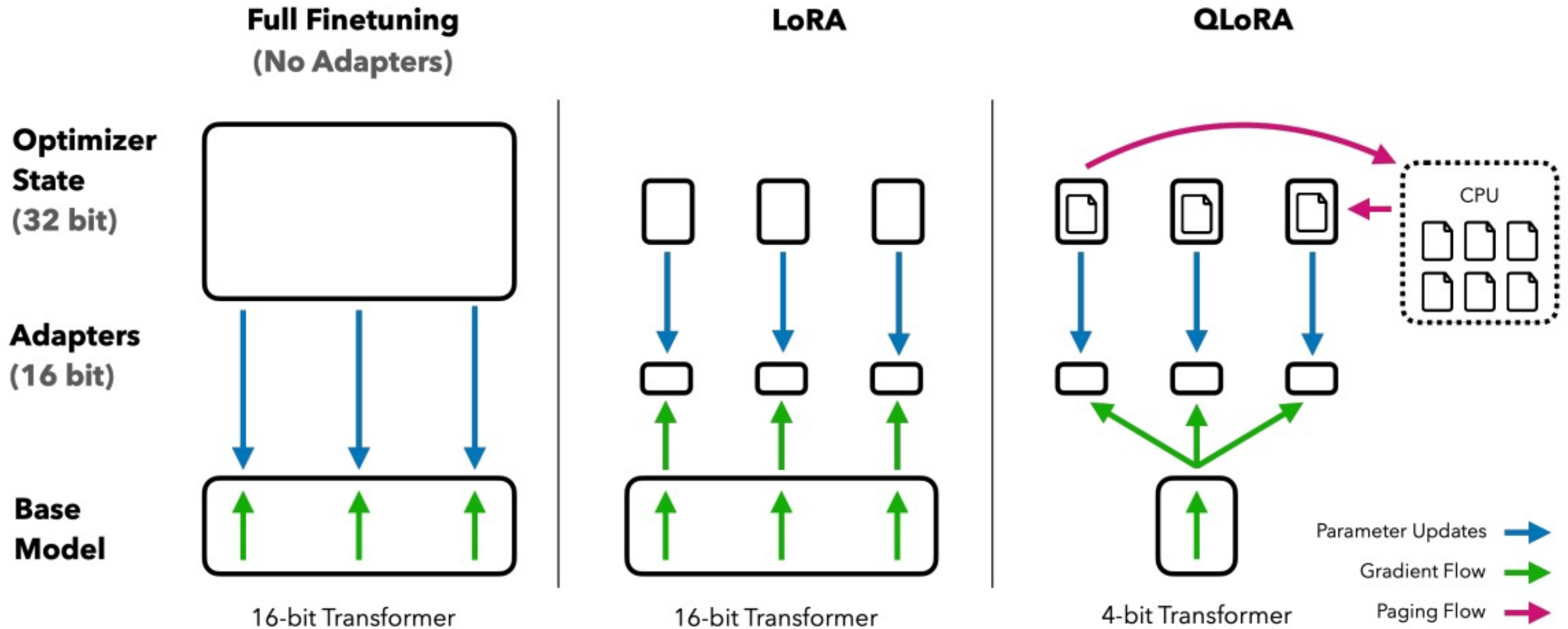


<https://huggingface.co/tasks>

Llama-2-chat uses RLHF to ensure safety and helpfulness



QLoRA: Efficient Finetuning of Quantized LLMs



QLoRA: Efficient Finetuning of Quantized LLMs

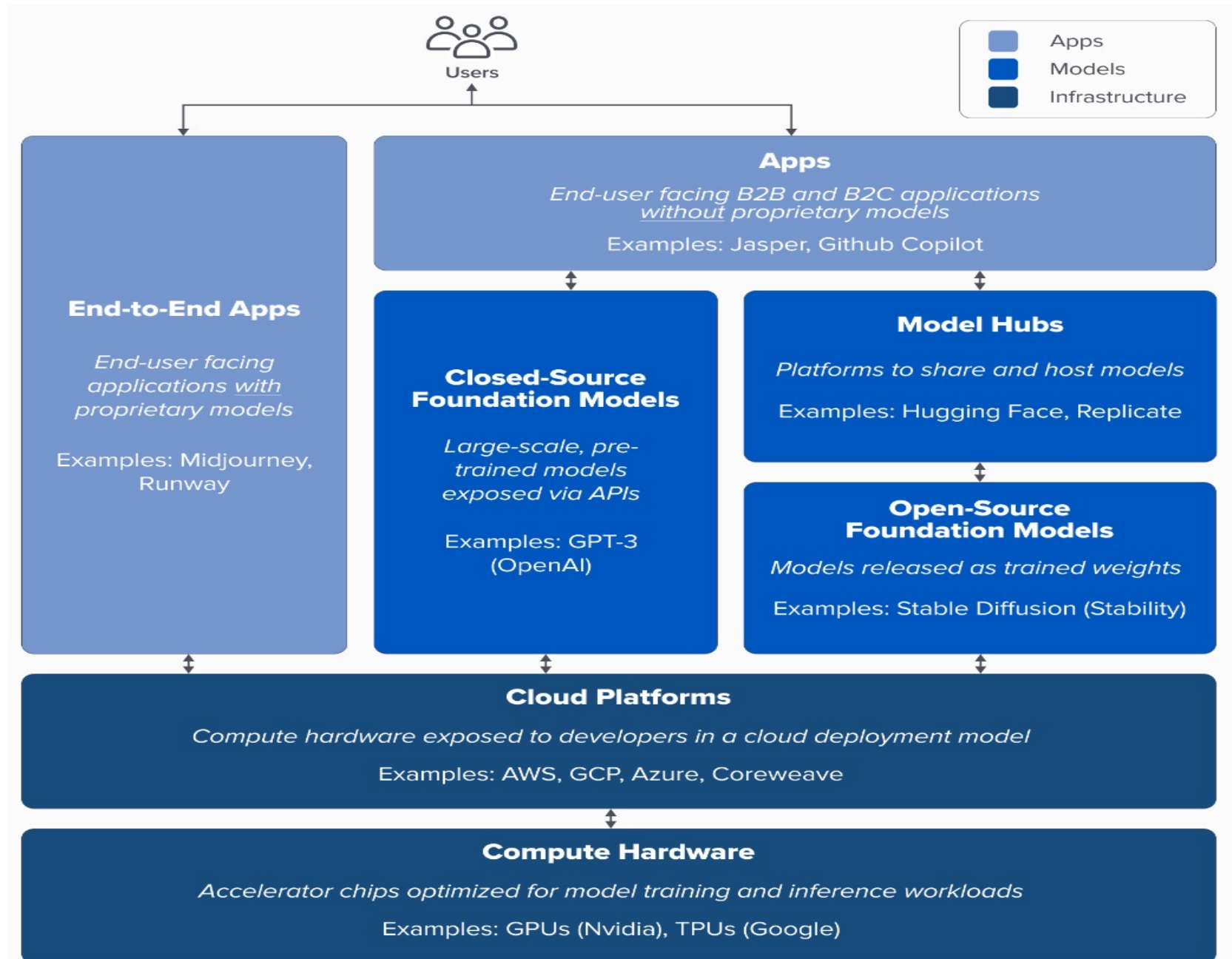
QLoRA reduces the average memory requirements of finetuning a **65B** parameter model from >780GB of **GPU memory** to **<48GB**

Model	Size	Elo
GPT-4	-	1348 ± 1
Guanaco 65B	41 GB	1022 ± 1
Guanaco 33B	21 GB	992 ± 1
Vicuna 13B	26 GB	974 ± 1
ChatGPT	-	966 ± 1
Guanaco 13B	10 GB	916 ± 1
Bard	-	902 ± 1
Guanaco 7B	6 GB	879 ± 1

QLoRA: Efficient Finetuning of Quantized LLMs

Model / Dataset	Params	Model bits	Memory	ChatGPT vs Sys	Sys vs ChatGPT	Mean	95% CI
GPT-4	-	-	-	119.4%	110.1%	114.5%	2.6%
Bard	-	-	-	93.2%	96.4%	94.8%	4.1%
Guanaco	65B	4-bit	41 GB	96.7%	101.9%	99.3%	4.4%
Alpaca	65B	4-bit	41 GB	63.0%	77.9%	70.7%	4.3%
FLAN v2	65B	4-bit	41 GB	37.0%	59.6%	48.4%	4.6%
Guanaco	33B	4-bit	21 GB	96.5%	99.2%	97.8%	4.4%
Open Assistant	33B	16-bit	66 GB	91.2%	98.7%	94.9%	4.5%
Alpaca	33B	4-bit	21 GB	67.2%	79.7%	73.6%	4.2%
FLAN v2	33B	4-bit	21 GB	26.3%	49.7%	38.0%	3.9%
Vicuna	13B	16-bit	26 GB	91.2%	98.7%	94.9%	4.5%
Guanaco	13B	4-bit	10 GB	87.3%	93.4%	90.4%	5.2%
Alpaca	13B	4-bit	10 GB	63.8%	76.7%	69.4%	4.2%
HH-RLHF	13B	4-bit	10 GB	55.5%	69.1%	62.5%	4.7%
Unnatural Instr.	13B	4-bit	10 GB	50.6%	69.8%	60.5%	4.2%
Chip2	13B	4-bit	10 GB	49.2%	69.3%	59.5%	4.7%
Longform	13B	4-bit	10 GB	44.9%	62.0%	53.6%	5.2%
Self-Instruct	13B	4-bit	10 GB	38.0%	60.5%	49.1%	4.6%
FLAN v2	13B	4-bit	10 GB	32.4%	61.2%	47.0%	3.6%
Guanaco	7B	4-bit	5 GB	84.1%	89.8%	87.0%	5.4%
Alpaca	7B	4-bit	5 GB	57.3%	71.2%	64.4%	5.0%
FLAN v2	7B	4-bit	5 GB	33.3%	56.1%	44.8%	4.0%

Generative AI Tech Stack



Generative AI Software and Business Factors

Business
Factors

Distribution

Proprietary Data

Domain Expertise

...

Application

A product utilizing and managing model inputs and outputs

Models

Large language models, image generation, or other ML models

Software

Data

Labeling, evaluation

MLOps Model management, tracking

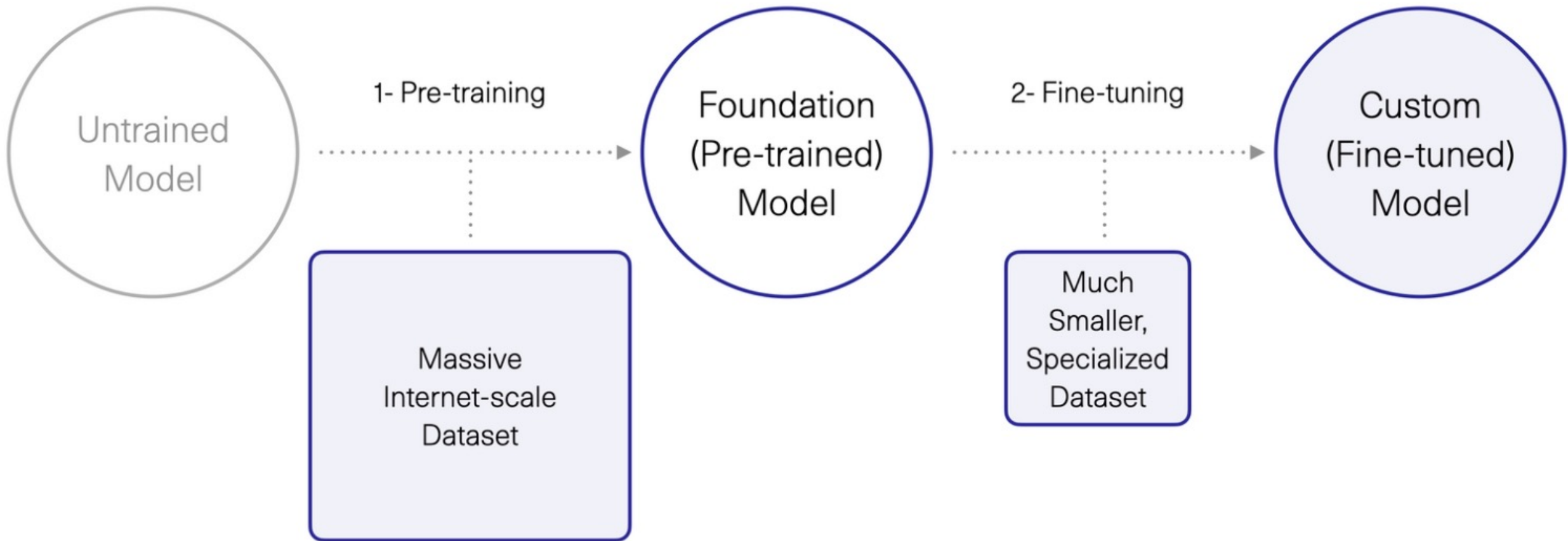
Cloud Platform

Hosting, compute, model deployment and monitoring

Generative AI

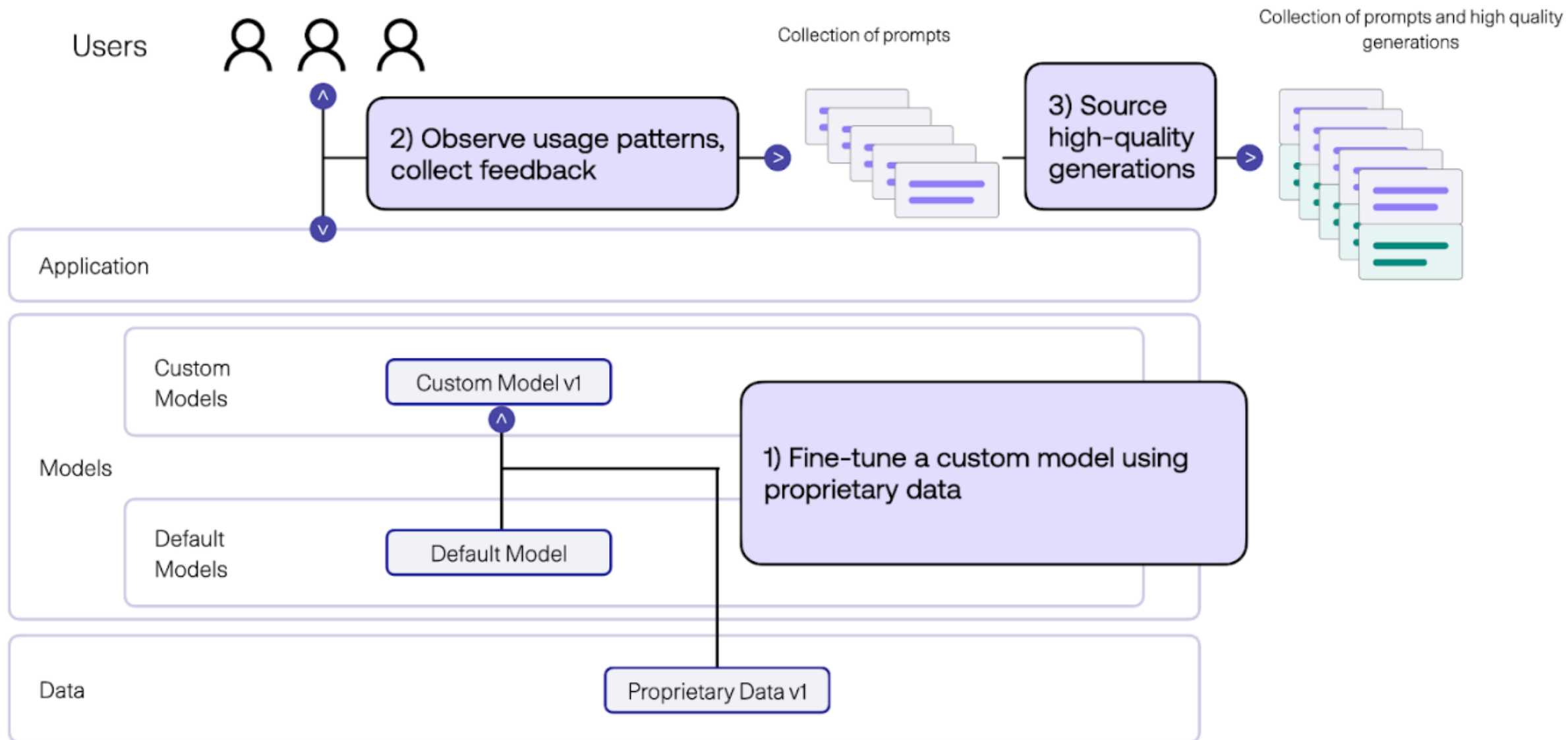
1. Pre-training Foundation (Pre-trained) Model

2. Fine-tuning Custom (Fine-tuned) Model



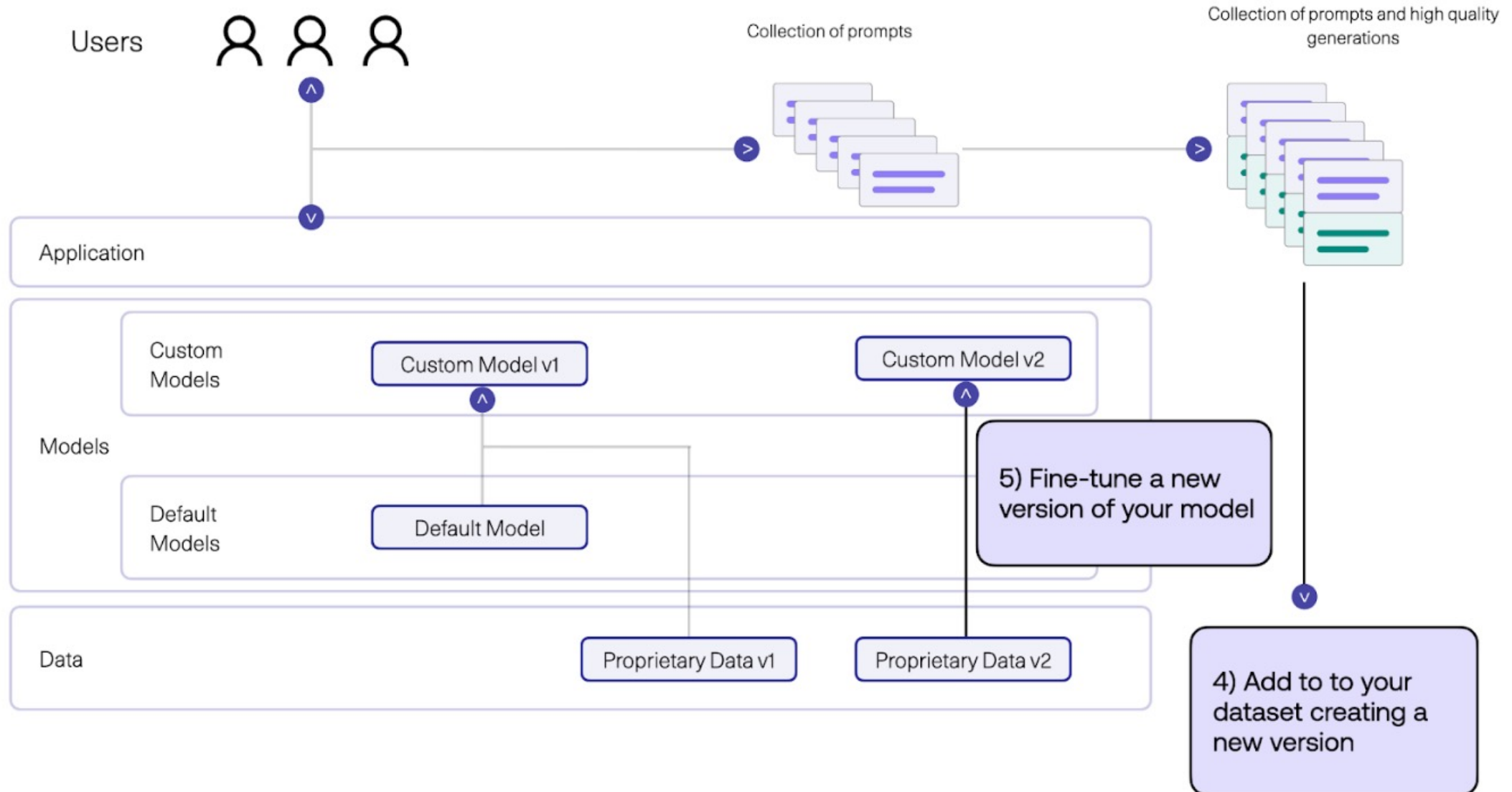
Generative AI

Fine-tune Custom Models using Proprietary Data

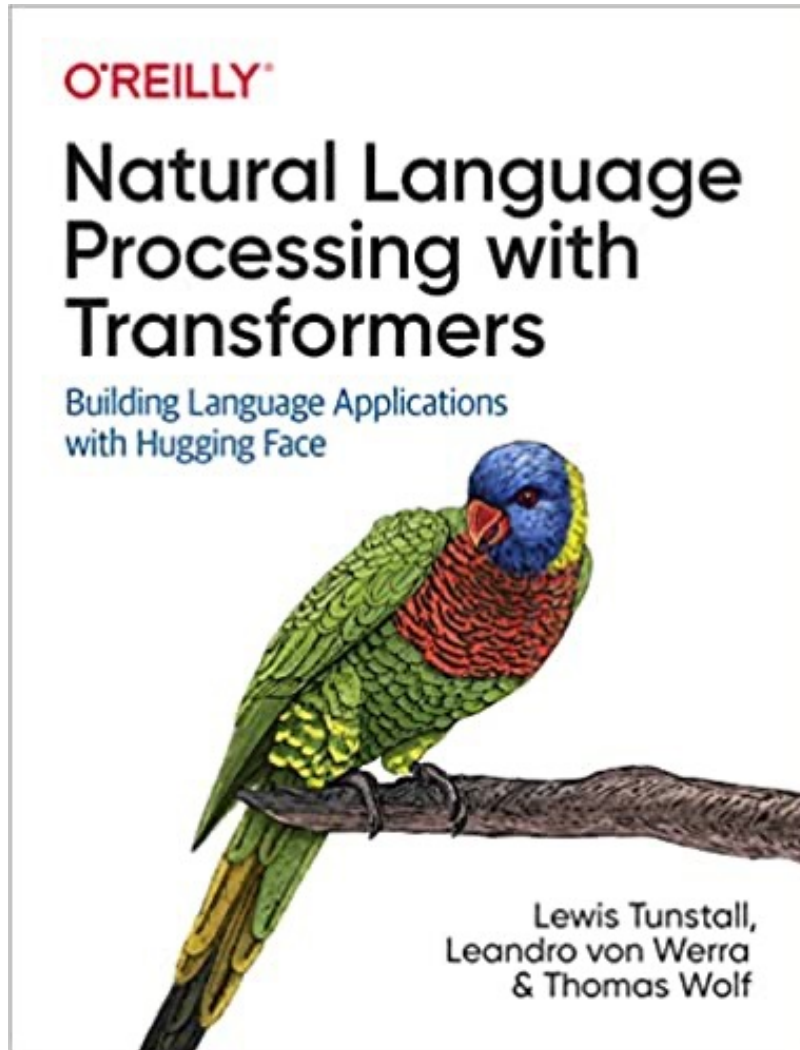


Generative AI

Fine-tune Custom Models using Proprietary Data



NLP with Transformers Github Notebooks



Running on a cloud platform

To run these notebooks on a cloud platform, just click on one of the badges in the table below:

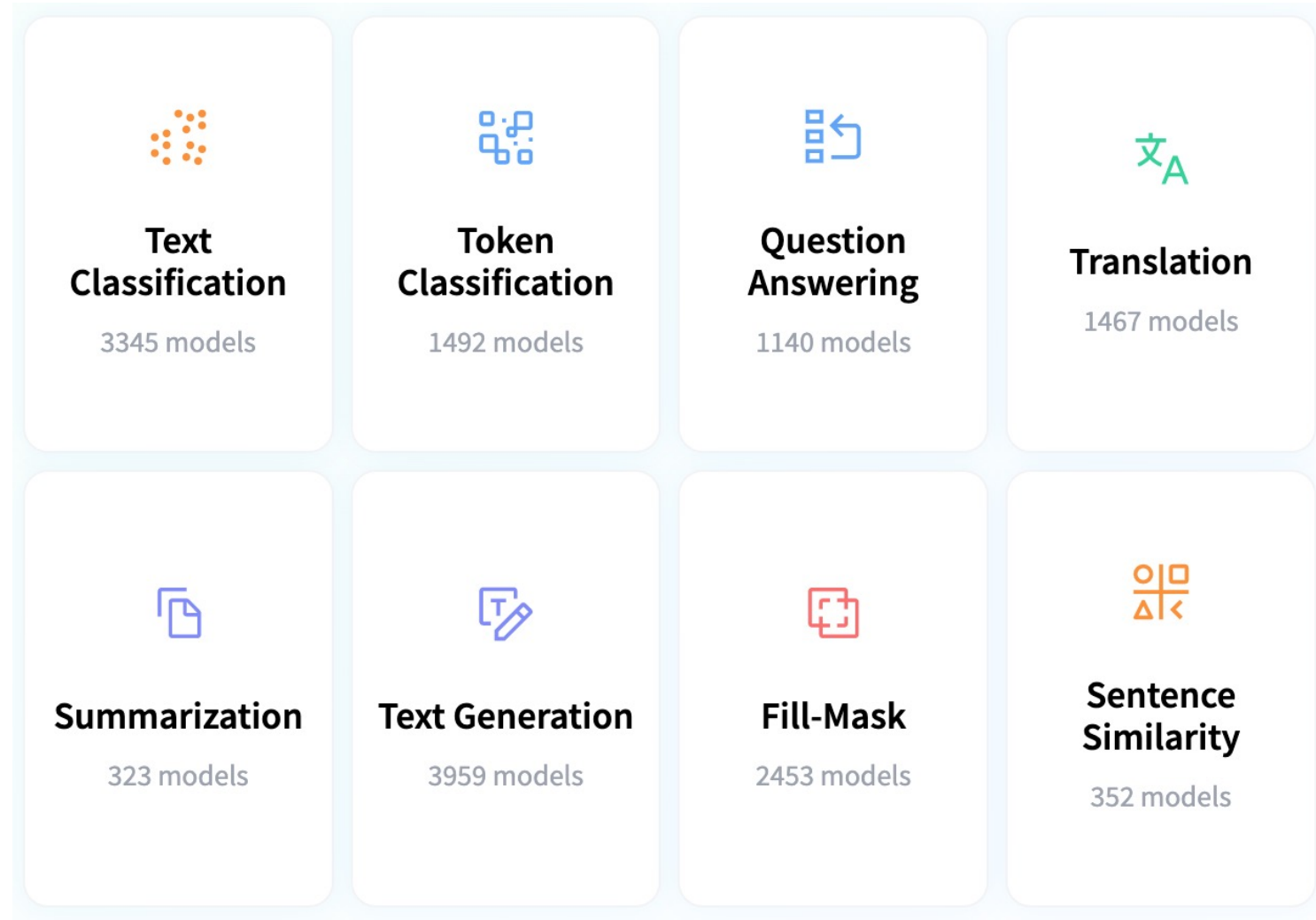
Chapter	Colab	Kaggle	Gradient	Studio Lab
Introduction	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Classification	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Transformer Anatomy	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Multilingual Named Entity Recognition	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Generation	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Summarization	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Question Answering	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Making Transformers Efficient in Production	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Dealing with Few to No Labels	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Training Transformers from Scratch	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Future Directions	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab

Nowadays, the GPUs on Colab tend to be K80s (which have limited memory), so we recommend using [Kaggle](#), [Gradient](#), or [SageMaker Studio Lab](#). These platforms tend to provide more performant GPUs like P100s, all for free!

<https://github.com/nlp-with-transformers/notebooks>

Hugging Face Tasks

Natural Language Processing



<https://huggingface.co/tasks>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

NLP with Transformers

Comment Share Settings

RAM Disk Editing

Table of contents

- Natural Language Processing with Transformers
 - Text Classification
 - Named Entity Recognition
 - Question Answering
 - Summarization
 - Translation
 - Text Generation
- AI in Finance
 - Normative Finance and Financial Theories
 - Uncertainty and Risk
 - Expected Utility Theory (EUT)
 - Mean-Variance Portfolio Theory (MVPT)
 - Capital Asset Pricing Model (CAPM)
 - Arbitrage Pricing Theory (APT)
 - Data Driven Finance
 - Financial Econometrics and Regression
 - Data Availability
 - Normative Theories Revisited
 - Mean-Variance Portfolio Theory

Natural Language Processing with Transformers

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

```
[1] 1 !git clone https://github.com/nlp-with-transformers/notebooks.git
    2 %cd notebooks
    3 from install import *
    4 install_requirements()
```

```
[3] 1 from utils import *
    2 setup_chapter()
```

```
[12] 1 text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
     2 from your online store in Germany. Unfortunately, when I opened the package, \
     3 I discovered to my horror that I had been sent an action figure of Megatron \
     4 instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
     5 dilemma. To resolve the issue, I demand an exchange of Megatron for the \
     6 Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
     7 this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

Text Classification

```
[13] 1 from transformers import pipeline
     2 classifier = pipeline("text-classification")
```

```
[14] 1 import pandas as pd
     2 outputs = classifier(text)
     3 pd.DataFrame(outputs)
```

<https://tinyurl.com/aintpupython101>

NLP with Transformers

```
!git clone https://github.com/nlp-with-transformers/notebooks.git
%cd notebooks
from install import *
install_requirements()
```

```
from utils import *
setup_chapter()
```

Text Classification

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

Text Classification

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

```
from transformers import pipeline
classifier = pipeline("text-classification")
```

```
import pandas as pd
outputs = classifier(text)
pd.DataFrame(outputs)
```

	label	score
0	NEGATIVE	0.901546

Text Classification

```
from transformers import pipeline  
classifier = pipeline("text-classification")
```

```
import pandas as pd  
outputs = classifier(text)  
pd.DataFrame(outputs)
```

	label	score
0	NEGATIVE	0.901546

Named Entity Recognition

```
ner_tagger = pipeline("ner", aggregation_strategy="simple")
outputs = ner_tagger(text)
pd.DataFrame(outputs)
```

	entity_group	score	word	start	end
0	ORG	0.879010	Amazon	5	11
1	MISC	0.990859	Optimus Prime	36	49
2	LOC	0.999755	Germany	90	97
3	MISC	0.556570	Mega	208	212
4	PER	0.590256	##tron	212	216
5	ORG	0.669692	Decept	253	259
6	MISC	0.498349	##icons	259	264
7	MISC	0.775362	Megatron	350	358
8	MISC	0.987854	Optimus Prime	367	380
9	PER	0.812096	Bumblebee	502	511

Question Answering

```
reader = pipeline("question-answering")
question = "What does the customer want?"
outputs = reader(question=question, context=text)
pd.DataFrame([outputs])
```

	score	start	end	answer
0	0.631292	335	358	an exchange of Megatron

Summarization

```
summarizer = pipeline("summarization")
outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])
```

Bumblebee ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead.

Text Summarization

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

```
from transformers import pipeline
summarizer = pipeline("summarization")
outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])
```

Bumblebee ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead.

Translation

```
translator = pipeline("translation_en_to_de",  
                      model="Helsinki-NLP/opus-mt-en-de")  
outputs = translator(text, clean_up_tokenization_spaces=True, min_length=100)  
print(outputs[0]['translation_text'])
```

Sehr geehrter Amazon, letzte Woche habe ich eine Optimus Prime Action Figur aus Ihrem Online-Shop in Deutschland bestellt. Leider, als ich das Paket öffnete, entdeckte ich zu meinem Entsetzen, dass ich stattdessen eine Action Figur von Megatron geschickt worden war! Als lebenslanger Feind der Decepticons, Ich hoffe, Sie können mein Dilemma verstehen. Um das Problem zu lösen, Ich fordere einen Austausch von Megatron für die Optimus Prime Figur habe ich bestellt. Anbei sind Kopien meiner Aufzeichnungen über diesen Kauf. Ich erwarte, bald von Ihnen zu hören. Aufrichtig, Bumblebee.

Text Generation

```
from transformers import set_seed
set_seed(42) # Set the seed to get reproducible results
```

```
generator = pipeline("text-generation")
response = "Dear Bumblebee, I am sorry to hear that your order was mixed up."
prompt = text + "\n\nCustomer service response:\n" + response
outputs = generator(prompt, max_length=200)
print(outputs[0]['generated_text'])
```

Customer service response:

Dear Bumblebee, I am sorry to hear that your order was mixed up. The order was completely mislabeled, which is very common in our online store, but I can appreciate it because it was my understanding from this site and our customer service of the previous day that your order was not made correct in our mind and that we are in a process of resolving this matter. We can assure you that your order

Text Generation

Dear Amazon, last week I ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead! As a lifelong enemy of the Decepticons, I hope you can understand my dilemma. To resolve the issue, I demand an exchange of Megatron for the Optimus Prime figure I ordered. Enclosed are copies of my records concerning this purchase. I expect to hear from you soon. Sincerely, Bumblebee.

Customer service response:

Dear Bumblebee, I am sorry to hear that your order was mixed up. The order was completely mislabeled, which is very common in our online store, but I can appreciate it because it was my understanding from this site and our customer service of the previous day that your order was not made correct in our mind and that we are in a process of resolving this matter. We can assure you that your order

Question Answering

```
!pip install transformers
from transformers import pipeline
qamodel = pipeline("question-answering")
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
qamodel(question = question, context = context)
```

```
{'answer': 'Taipei', 'end': 39, 'score': 0.9730741381645203, 'start': 33}
```

Question Answering

```
from transformers import pipeline
qamodel = pipeline("question-answering", model='deepset/roberta-base-squad2')
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
output = qamodel(question = question, context = context)
print(output['answer'])
```

Taipei

Text Generation with LLM (zephyr-7b-beta)

```
# Install transformers from source - only needed for versions <= v4.34
!pip install git+https://github.com/huggingface/transformers.git
!pip install accelerate
```

Text Generation with LLM

```
import torch
from transformers import pipeline

pipe = pipeline("text-generation",
model="HuggingFaceH4/zephyr-7b-beta",
torch_dtype=torch.bfloat16,
device_map="auto")
```

Text Generation with LLM

```
# We use the tokenizer's chat template to format each message - see
https://huggingface.co/docs/transformers/main/en/chat_templating
messages = [
    {
        "role": "system",
        "content": "You are a friendly chatbot who always responds in the style of a pirate",
    },
    {"role": "user", "content": "How many helicopters can a human eat in one sitting?"},
]
prompt = pipe.tokenizer.apply_chat_template(messages,
tokenize=False, add_generation_prompt=True)

outputs = pipe(prompt, max_new_tokens=256,
do_sample=True, temperature=0.7, top_k=50, top_p=0.95)
print(outputs[0]["generated_text"])
```

Text Generation with LLM

```
import torch
from transformers import pipeline

pipe = pipeline("text-generation", model="HuggingFaceH4/zephyr-7b-beta",
torch_dtype=torch.bfloat16, device_map="auto")

# We use the tokenizer's chat template to format each message - see
https://huggingface.co/docs/transformers/main/en/chat_templating
messages = [
    {
        "role": "system",
        "content": "You are a friendly chatbot who always responds in the style of a pirate",
    },
    {"role": "user", "content": "How many helicopters can a human eat in one sitting?"},
]

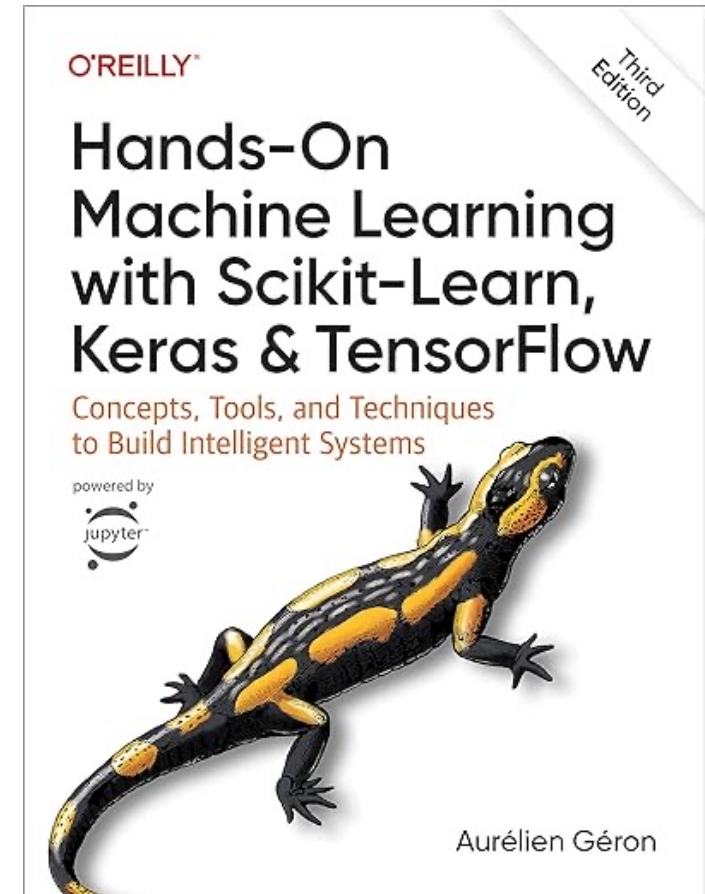
prompt = pipe.tokenizer.apply_chat_template(messages, tokenize=False,
add_generation_prompt=True)
outputs = pipe(prompt, max_new_tokens=256, do_sample=True, temperature=0.7,
top_k=50, top_p=0.95)
print(outputs[0]["generated_text"])
```

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

Notebooks

- [1. The Machine Learning landscape](#)
- [2. End-to-end Machine Learning project](#)
- [3. Classification](#)
- [4. Training Models](#)
- [5. Support Vector Machines](#)
- [6. Decision Trees](#)
- [7. Ensemble Learning and Random Forests](#)
- [8. Dimensionality Reduction](#)
- [9. Unsupervised Learning Techniques](#)
- [10. Artificial Neural Nets with Keras](#)
- [11. Training Deep Neural Networks](#)
- [12. Custom Models and Training with TensorFlow](#)
- [13. Loading and Preprocessing Data](#)
- [14. Deep Computer Vision Using Convolutional Neural Networks](#)**
- [15. Processing Sequences Using RNNs and CNNs](#)**
- [16. Natural Language Processing with RNNs and Attention](#)**
- [17. Autoencoders, GANs, and Diffusion Models](#)
- [18. Reinforcement Learning](#)**
- [19. Training and Deploying TensorFlow Models at Scale](#)

<https://github.com/ageron/handson-ml3>



Papers with Code State-of-the-Art (SOTA)



Search for papers, code and tasks



[Browse State-of-the-Art](#)

[Follow](#)

[Discuss](#)

[Trends](#)

[About](#)

[Log In/Register](#)

Browse State-of-the-Art

1509 leaderboards • 1327 tasks • 1347 datasets • 17810 papers with code

Follow on [Twitter](#) for updates

Computer Vision



Semantic Segmentation

33 leaderboards
667 papers with code



Image Classification

52 leaderboards
564 papers with code



Object Detection

54 leaderboards
467 papers with code



Image Generation

51 leaderboards
231 papers with code



Pose Estimation

40 leaderboards
231 papers with code

[See all 707 tasks](#)

Natural Language Processing



Machine Translation



Language Modelling



Question Answering



Sentiment Analysis



Text Generation

<https://paperswithcode.com/sota>

Summary

- **Text Analytics with Python**
- **Large Language Models (LLMs)**
- **Generative AI**

References

- Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson.
- Numa Dhamani and Maggie Engler (2024), Introduction to Generative AI, Manning
- Denis Rothman (2024), Transformers for Natural Language Processing and Computer Vision - Third Edition: Explore Generative AI and Large Language Models with Hugging Face, ChatGPT, GPT-4V, and DALL-E 3, 3rd ed. Edition, Packt Publishing
- Ben Auffarth (2023), Generative AI with LangChain: Build large language model (LLM) apps with Python, ChatGPT and other LLMs, Packt Publishing.
- Aurélien Géron (2022), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 3rd Edition, O'Reilly Media.
- Steven D'Ascoli (2022), Artificial Intelligence and Deep Learning with Python: Every Line of Code Explained For Readers New to AI and New to Python, Independently published.
- Nithin Buduma, Nikhil Buduma, Joe Papa (2022), Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms, 2nd Edition, O'Reilly Media.
- Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. (2024) "Reasoning with Large Language Models, a Survey." arXiv preprint arXiv:2407.11511.
- Line of Code Explained For Readers New to AI and New to Python, Independently published.
- Steven D'Ascoli (2022), Artificial Intelligence and Deep Learning with Python: Every Line of Code Explained For Readers New to AI and New to Python, Independently published.
- Dipanjan Sarkar (2019), Text Analytics with Python: A Practitioner's Guide to Natural Language Processing, Second Edition. APress. <https://github.com/Apress/text-analytics-w-python-2e>
- Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda (2018), Applied Text Analysis with Python, O'Reilly Media. <https://www.oreilly.com/library/view/applied-text-analysis/9781491963036/>
- Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, Ray Kurzweil (2018). Universal Sentence Encoder. arXiv:1803.11175.
- Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-hsuan Sung, Ray Kurzweil (2019). Multilingual Universal Sentence Encoder for Semantic Retrieval.
- Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang (2020). "Pre-trained Models for Natural Language Processing: A Survey." arXiv preprint arXiv:2003.08271.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.
- Jay Alammar (2019), The Illustrated Transformer, <http://jalammar.github.io/illustrated-transformer/>
- Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, <http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/>
- Christopher Olah, (2015) Understanding LSTM Networks, <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- HuggingFace (2020), Transformers Notebook, <https://huggingface.co/transformers/notebooks.html>
- The Super Duper NLP Repo, <https://notebooks.quantumstat.com/>
- Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer (2020). "Deep learning for financial applications: A survey." Applied Soft Computing (2020): 106384.
- Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu (2020), "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019." Applied Soft Computing 90 (2020): 106181.
- Deep Learning Basics: Neural Networks Demystified, <https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU>
- Deep Learning SIMPLIFIED, <https://www.youtube.com/playlist?list=PLjJh1vISEYgvGod9wWiydumYI8hOXixNu>
- 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, <https://www.youtube.com/watch?v=aircAruvnKk>
- 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, <https://www.youtube.com/watch?v=IHZwWFHwa-w>
- 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, <https://www.youtube.com/watch?v=Ilg3gGewQ5U>
- 3Blue1Brown (2024), Transformers (how LLMs work) explained visually | DL5, <https://www.youtube.com/watch?v=wjZofjX0v4M>
- 3Blue1Brown (2024), Attention in transformers, visually explained | DL6, <https://www.youtube.com/watch?v=eMlx5FFNoYc>
- 3Blue1Brown (2024), How might LLMs store facts | DL7, <https://www.youtube.com/watch?v=9-Jl0dxWQs8>
- 3Blue1Brown (2024), Large Language Models explained briefly, <https://www.youtube.com/watch?v=LPZh9BOjkQs>
- Min-Yuh Day (2024), Python 101, <https://tinyurl.com/aintpuython101>