

# CENG501 – Deep Learning

Week 8

Fall 2024

Sinan Kalkan

Dept. of Computer Engineering, METU

# Pre-training in NLP

Previously on CENG501

- Autoregressive language modeling
- Masked language modeling
- Next sentence prediction

Previously on GP-ENG501  
9-1

- 12 layer decoder-only transformer
- Unsupervised pretraining
  - BookCorpus dataset
- Supervised finetuning
  - Textual alignment
  - QA & commonsense reasoning
  - Semantic similarity
  - Classification

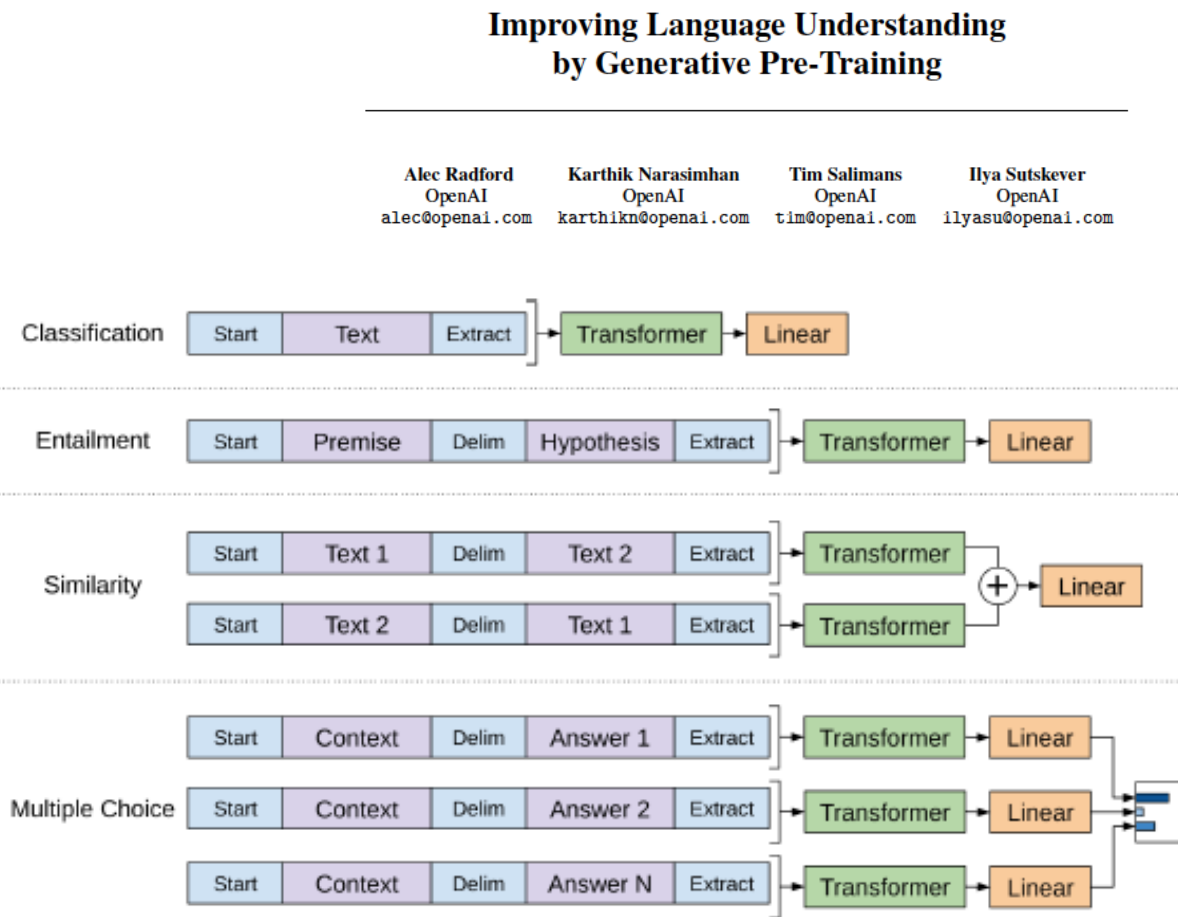
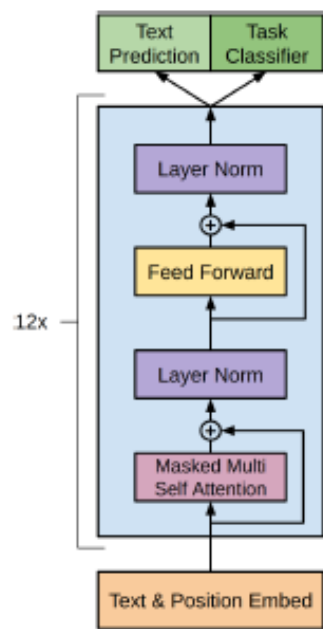
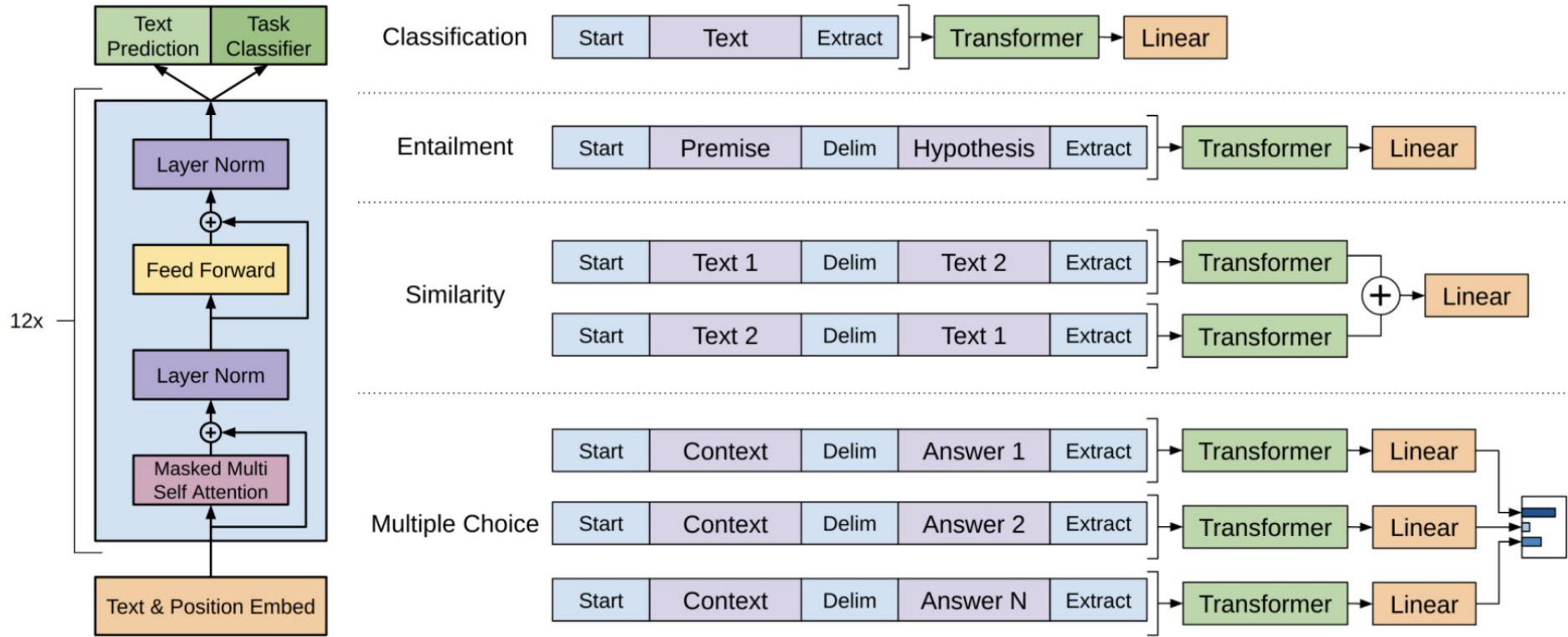


Figure 1: (left) Transformer architecture and training objectives used in this work. (right) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

# Discriminative Fine-tuning



# Previously on GP-CENG501

## GPT-1 Results

Table 2: Experimental results on natural language inference tasks, comparing our model with current state-of-the-art methods. 5x indicates an ensemble of 5 models. All datasets use accuracy as the evaluation metric.

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	<u>89.3</u>	-	-	-
CAFE [58] (5x)	80.2	79.0	<u>89.3</u>	-	-	-
Stochastic Answer Network [35] (3x)	<u>80.6</u>	<u>80.1</u>	-	-	-	-
CAFE [58]	78.7	77.9	88.5	<u>83.3</u>		
GenSen [64]	71.4	71.3	-	-	<u>82.3</u>	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	<b>61.7</b>
Finetuned Transformer LM (ours)	<b>82.1</b>	<b>81.4</b>	<b>89.9</b>	<b>88.3</b>	<b>88.1</b>	56.0

BERT  
Previously on CENG501

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova

Google AI Language

{jacobdevlin, mingweichang, kentonl, kristout}@google.com

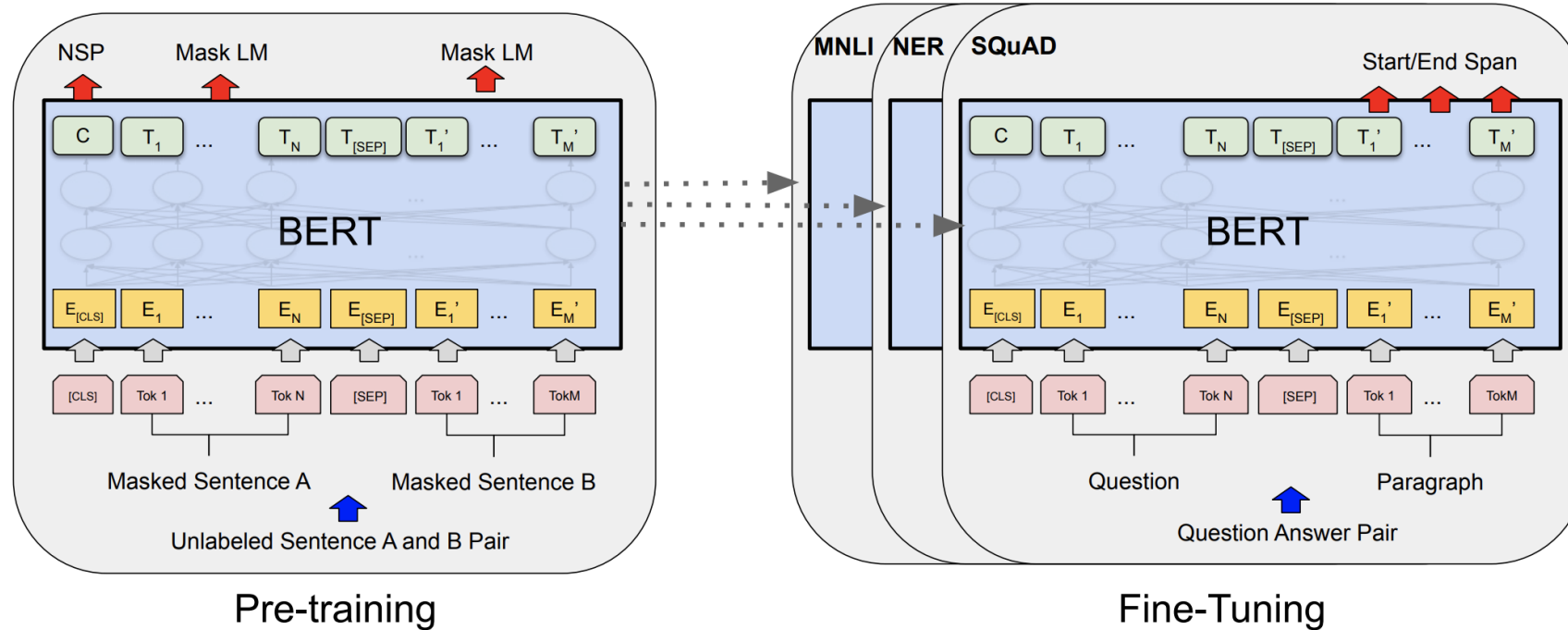
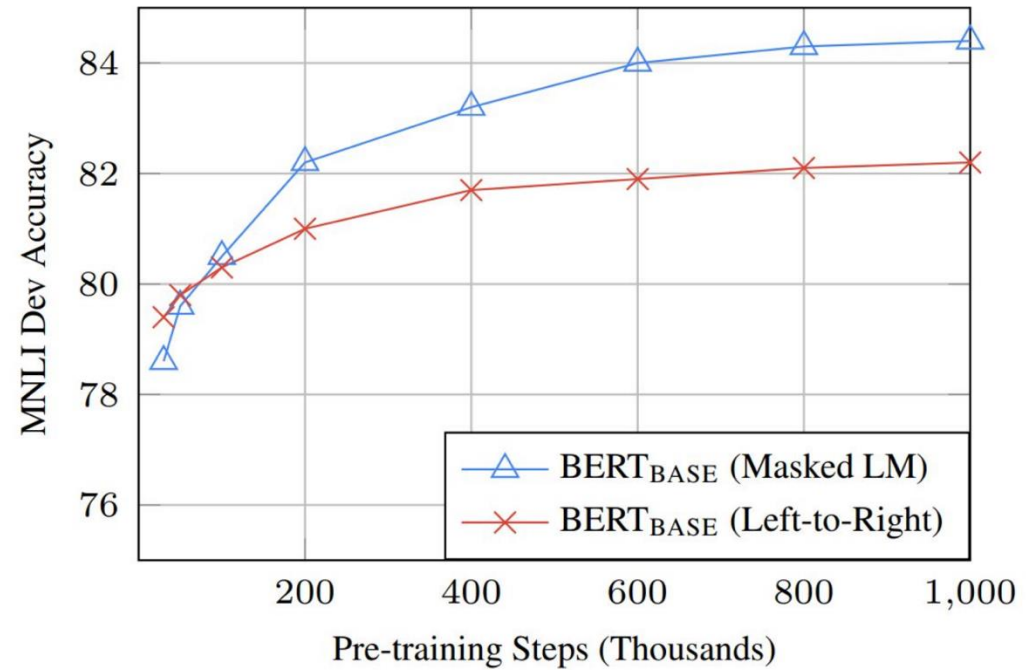
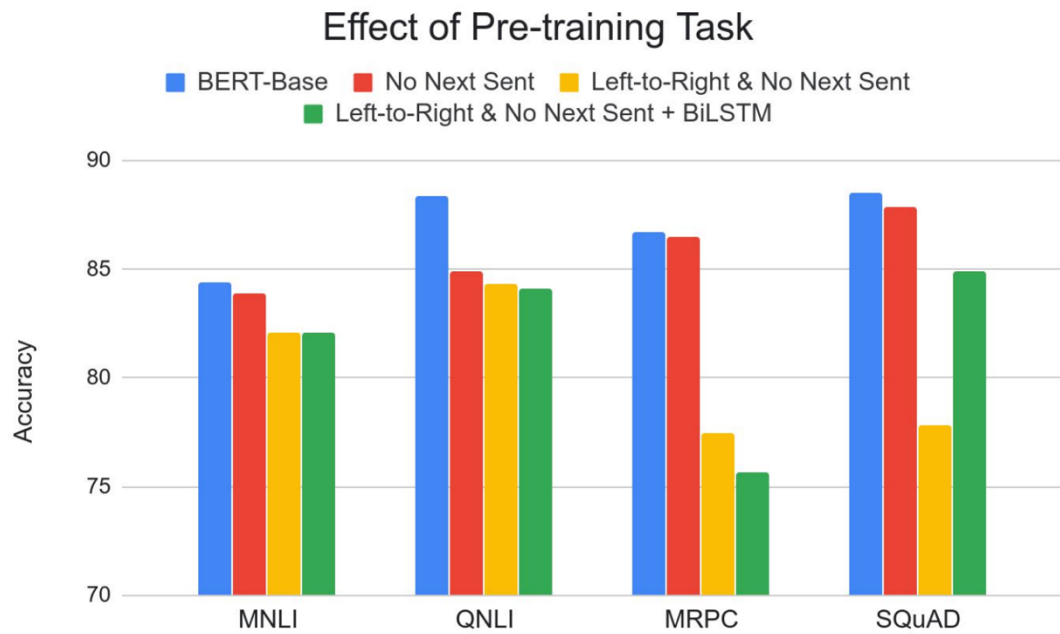


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

**BERT**  
Previously on CENG501



Previously on CENG501

Model	Title	Focus	Paradigm	Params
GPT-1	Improving <u>Language Understanding</u> by <u>Generative Pre-Training</u>	NLU tasks, pre-trained model	Pre-training->Efficient Fine-tuning	117M
GPT-2	Language Models are <u>Unsupervised Multitask Learners</u>	Zero-shot Evaluation, NLG Tasks	Pre-training->Zero-shot Multitask Transfer	1.5B
GPT-3	Language Models are <u>Few-Shot Learners</u>	Few-shot Learning or In-context Learning	In-context Learning with a few demonstration examples	175B
GPT-3.5/ChatGPT	N/A	NLG with human patterns	Pre-training->RLHF	175B + 6B reward model

- GPT is out before BERT.

Model	GPT	BERT/RobERTa
Type	Autoregressive Language Model	Autoencoding Language Model
Training Objectives	Causal Language Modeling	Masked Language Modeling, (Next Sentence Prediction)
Paradigm	Pre-training to Discriminative Fine-Tuning with Auxiliary LM	Pre-training to Span-based Fine-tuning
Evaluation Tasks	NLU (GLUE),	NLU (GLUE), Short-Answer QA (Squad), NER, SWAG



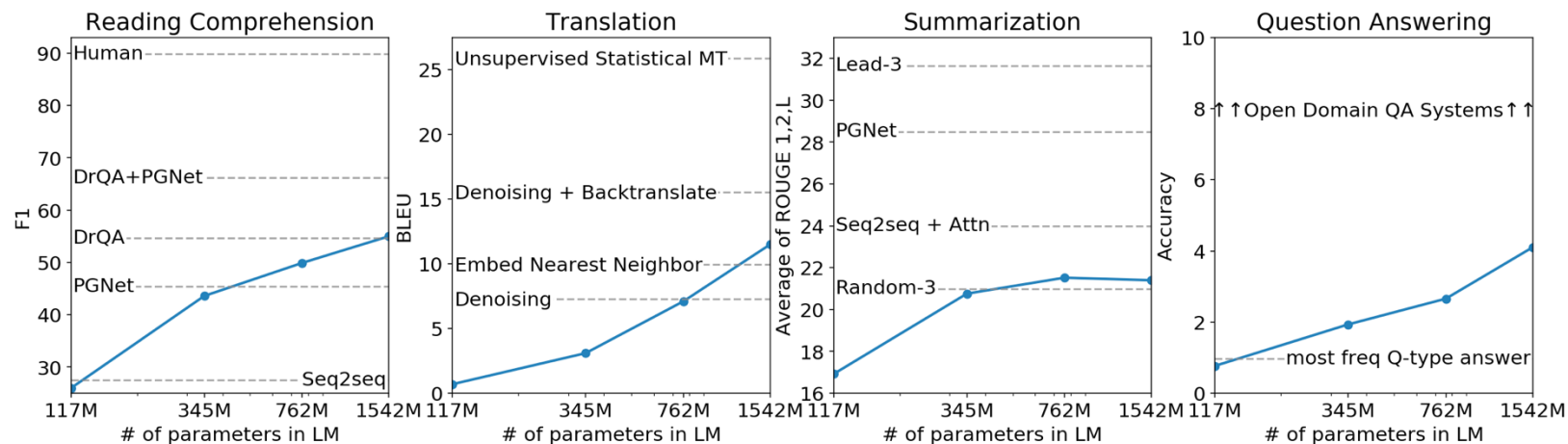
Previously on GPT-2  
GPT-2-2

## Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

## Language Models are Unsupervised Multitask Learners

Alec Radford<sup>\*1</sup> Jeffrey Wu<sup>\*1</sup> Rewon Child<sup>1</sup> David Luan<sup>1</sup> Dario Amodei<sup>\*\*1</sup> Ilya Sutskever<sup>\*\*1</sup>



- Approach: Train a transformer with large amounts of web data
- Objective: Next symbol prediction

symbols as the product of conditional probabilities (Jelinek & Mercer, 1980) (Bengio et al., 2003):

$$p(x) = \prod_{i=1}^n p(s_i | s_1, \dots, s_{i-1}) \quad (1)$$

This approach allows for tractable sampling from and estimation of  $p(x)$  as well as any conditionals of the form  $p(s_{n-k}, \dots, s_n | s_1, \dots, s_{n-k-1})$ . In recent years, there have

---

”I’m not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile** [I’m not a fool].

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: ”**Mentez mentez, il en restera toujours quelque chose,**” which translates as, ”**Lie lie and something will always remain.**”

“I hate the word ‘**perfume,**’” Burr says. ‘It’s somewhat better in French: ‘**parfum.**’

If listened carefully at 29:55, a conversation can be heard between two guys in French: “-**Comment on fait pour aller de l’autre coté? -Quel autre coté?**”, which means “- **How do you get to the other side? - What side?**”.

If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?

“**Brevet Sans Garantie Du Gouvernement**”, translated to English: “**Patented without government warranty**”.

---

*Table 1.* Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.

## GPT-2: Language Modeling Benchamarks

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	<b>21.8</b>
117M	<b>35.13</b>	45.99	<b>87.65</b>	<b>83.4</b>	<b>29.41</b>	65.85	1.16	1.17	37.50	75.20
345M	<b>15.60</b>	55.48	<b>92.35</b>	<b>87.1</b>	<b>22.76</b>	47.33	1.01	<b>1.06</b>	26.37	55.72
762M	<b>10.87</b>	<b>60.12</b>	<b>93.45</b>	<b>88.0</b>	<b>19.93</b>	<b>40.31</b>	<b>0.97</b>	<b>1.02</b>	22.05	44.575
1542M	<b>8.63</b>	<b>63.24</b>	<b>93.30</b>	<b>89.05</b>	<b>18.34</b>	<b>35.76</b>	<b>0.93</b>	<b>0.98</b>	<b>17.48</b>	42.16

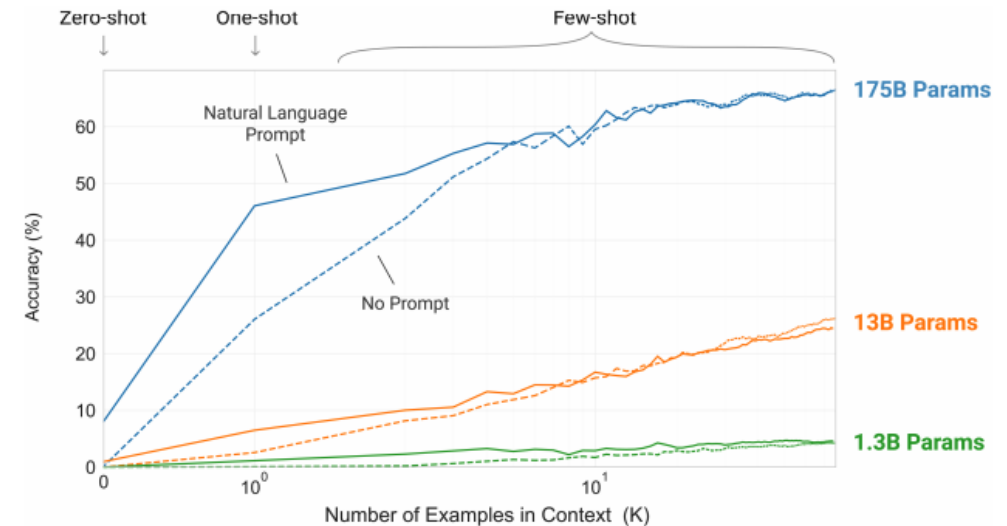
*Table 3.* Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

Previously on GP-ENG501  
 • 175B parameters!

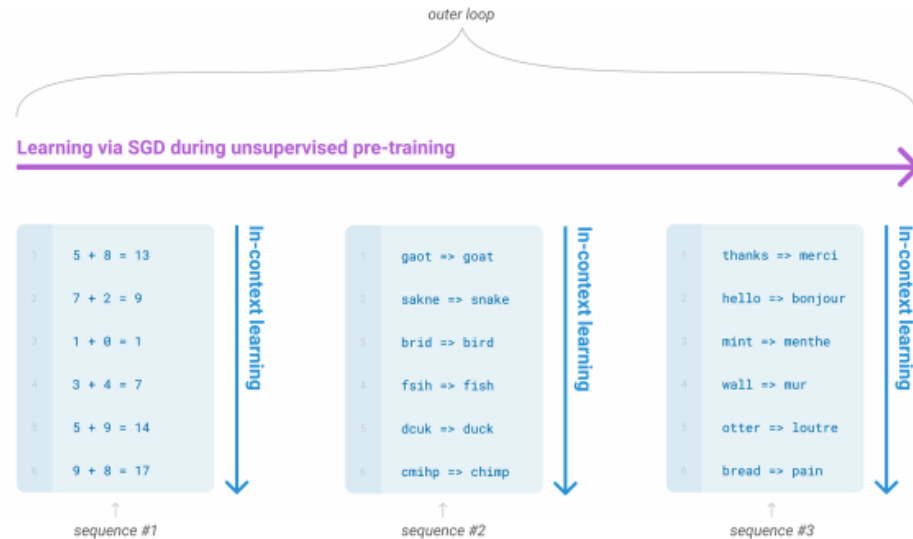
## Language Models are Few-Shot Learners

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

OpenAI



**Figure 1.2: Larger models make increasingly efficient use of in-context information.** We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper “in-context learning curves” for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.



**Figure 1.1: Language model meta-learning.** During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term “in-context learning” to describe the inner loop of this process, which occurs within the forward-pass upon each sequence. The sequences in this diagram are not intended to be representative of the data a model would see during pre-training, but are intended to show that there are sometimes repeated sub-tasks embedded within a single sequence.

## Three ways of in-context learning:

### Zero-shot

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

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

### One-shot

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

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

In a single sequence input, the prompted example can learn from previous demonstrations.

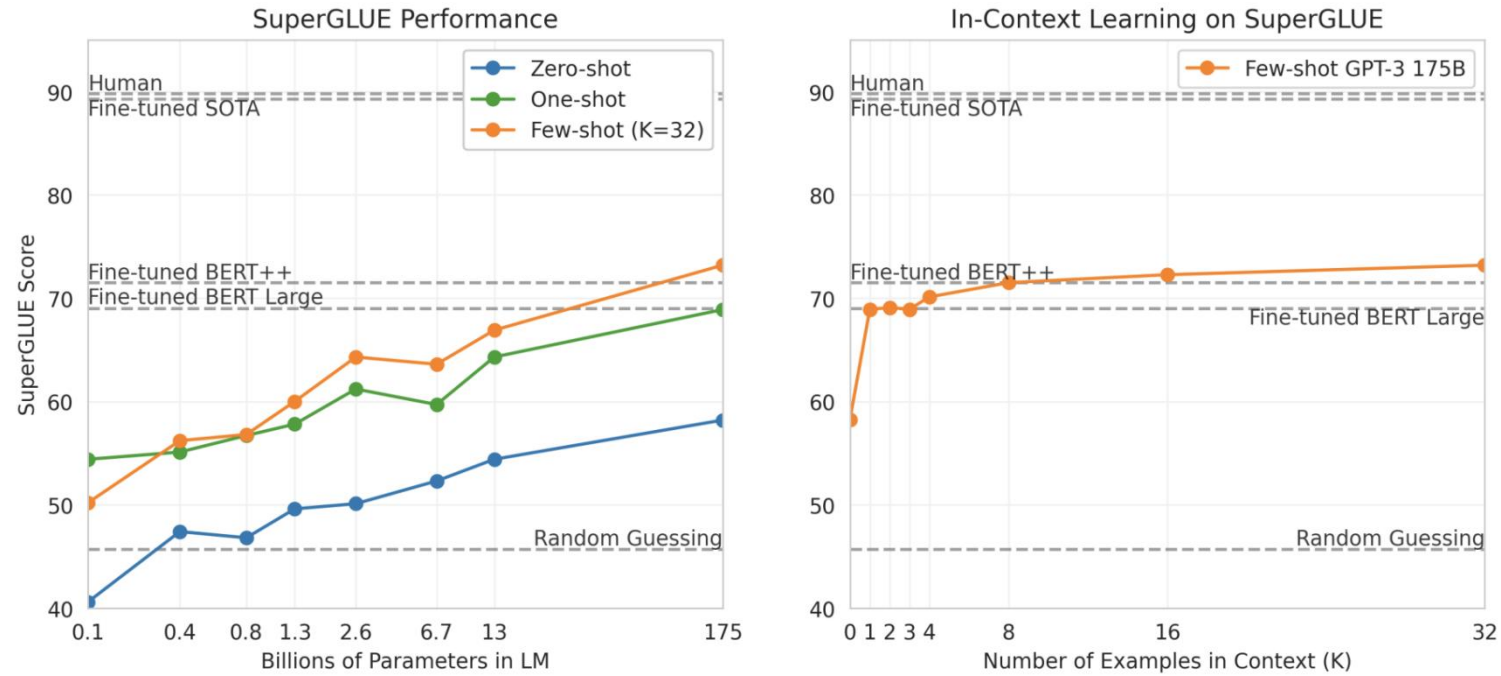
### Few-shot

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

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

Previously on CENG501

# GPT-3 Results: NLU of SuperGLUE



**Figure 3.8: Performance on SuperGLUE increases with model size and number of examples in context.** A value of  $K = 32$  means that our model was shown 32 examples per task, for 256 examples total divided across the 8 tasks in SuperGLUE. We report GPT-3 values on the dev set, so our numbers are not directly comparable to the dotted reference lines (our test set results are in Table 3.8). The BERT-Large reference model was fine-tuned on the SuperGLUE training set (125K examples), whereas BERT++ was first fine-tuned on MultiNLI (392K examples) and SWAG (113K examples) before further fine-tuning on the SuperGLUE training set (for a total of 630K fine-tuning examples). We find the difference in performance between the BERT-Large and BERT++ to be roughly equivalent to the difference between GPT-3 with one example per context versus eight examples per context.

Previously on CENG501

## Key to Success: Data Resources

Model	Pre-training Data	Size
GPT-1	BooksCorpus (7000 books)	5GB
BERT	BooksCorpus, En-Wikipedia	16GB
GPT-2	WebText	40GB
RoBERTa	BooksCorpus, CC-News, OpenWebText(WebText), Stories	160GB
GPT-3	CC(Common Crawl), WebText2, Books1, Books2, Wikipedia	~700GB
GPT-J	Pile Corpus	800GB

## Key to Success: Scaling Up

Model Name	$n_{\text{params}}$	$n_{\text{layers}}$	$d_{\text{model}}$	$n_{\text{heads}}$	$d_{\text{head}}$	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0 \times 10^{-4}$
GPT-3 Medium	350M	24	1024	16	64	0.5M	$3.0 \times 10^{-4}$
GPT-3 Large	760M	24	1536	16	96	0.5M	$2.5 \times 10^{-4}$
GPT-3 XL	1.3B	24	2048	24	128	1M	$2.0 \times 10^{-4}$
GPT-3 2.7B	2.7B	32	2560	32	80	1M	$1.6 \times 10^{-4}$
GPT-3 6.7B	6.7B	32	4096	32	128	2M	$1.2 \times 10^{-4}$
GPT-3 13B	13.0B	40	5140	40	128	2M	$1.0 \times 10^{-4}$
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	$0.6 \times 10^{-4}$

**Table 2.1:** Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.



Previously on CENG501

Step 1  
Collect demonstration data and train a supervised policy.

A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.

We give treats and punishments to teach...

SFT



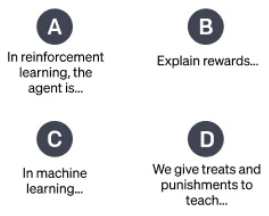
This data is used to fine-tune GPT-3.5 with supervised learning.



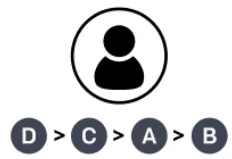
<https://openai.com/index/chatgpt/>

Step 2  
Collect comparison data and train a reward model.

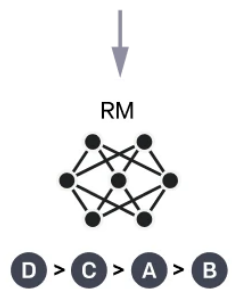
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



Sinan Kalkan

Step 3  
Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.



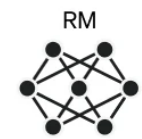
The PPO model is initialized from the supervised policy.



The policy generates an output.

Once upon a time...

The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

$r_k$



17

# Gemini: A Family of Highly Capable Multimodal Models

Gemini Team, Google<sup>1</sup>

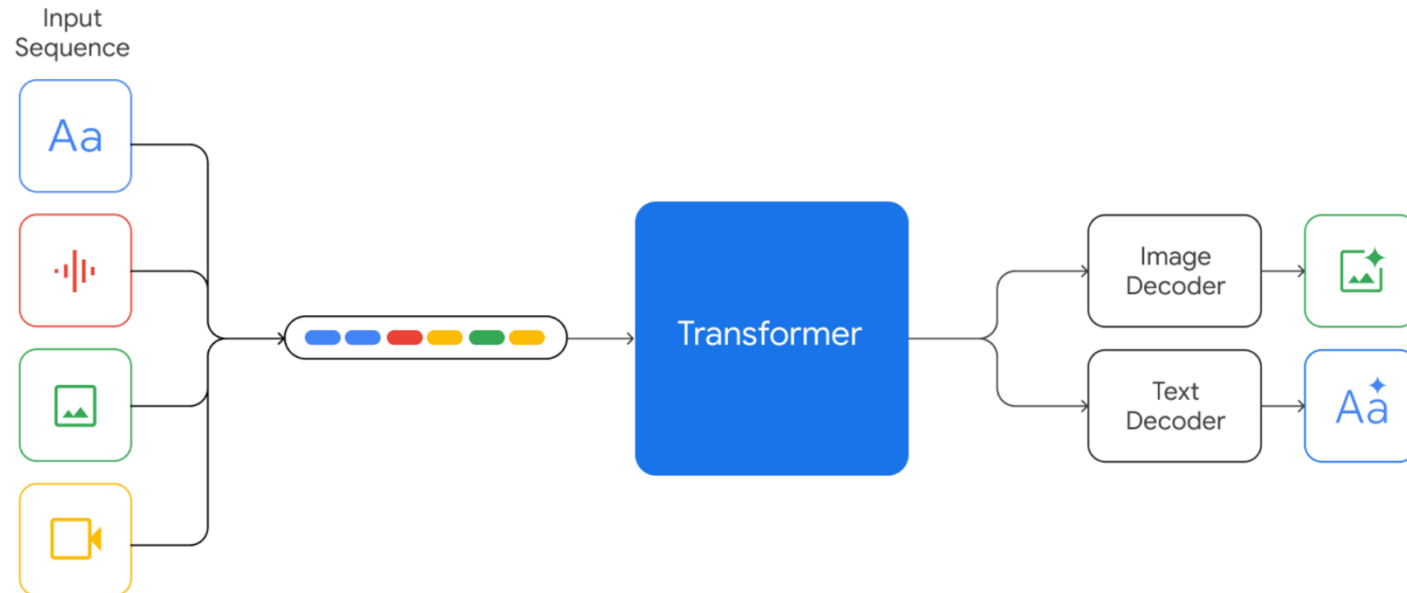


Figure 2 | Gemini models support interleaved sequences of text, image, audio, and video as inputs (illustrated by tokens of different colors in the input sequence). They can output responses with interleaved image and text.

# Other LLMs

Previously on CENG501

## 2 Approach

Our training approach is similar to the methods described in previous work (Brown et al., 2020; Chowdhery et al., 2022), and is inspired by the Chinchilla scaling laws (Hoffmann et al., 2022). We train large transformers on a large quantity of textual data using a standard optimizer.

## LLaMA: Open and Efficient Foundation Language Models

Hugo Touvron\*, Thibaut Lavril\*, Gautier Izacard\*, Xavier Martinet  
Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal  
Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin  
Edouard Grave\*, Guillaume Lample\*

Meta AI

2023

This paper was published in 2022. The main goal of this paper was to find the relationship between three factors. These factors are model size, number of tokens, and compute budget. They came to the conclusion that the current LLMs like 175B GPT-3, 280B Gopher, and 530B Megatron are significantly undertrained. All these models have increased the number of parameters but the training data remained constant. The authors mention that for compute-optimal training, the number of training tokens and model size must be scaled equally. They trained about 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens.

*After finding the relationship between the three factors, they trained a new LLM called Chinchilla which uses same compute budget as 280B Gopher but has 70B parameters and 4 times more training data. Chinchilla outperforms Gopher (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron (530B). This result is in contradiction to the “Scaling laws for LLMs” by OpenAI. Now, relatively smaller models can give better performance if trained on more data. Smaller models are easy to fine-tune and also have less latency at inference. These models should not be to their lowest possible loss to be compute optimal.*

Previously on CENG501

# Limits of LLMs

## Sparks of Artificial General Intelligence: Early experiments with GPT-4

Sébastien Bubeck    Varun Chandrasekaran    Ronen Eldan    Johannes Gehrke  
Eric Horvitz    Ece Kamar    Peter Lee    Yin Tat Lee    Yuanzhi Li    Scott Lundberg  
Harsha Nori    Hamid Palangi    Marco Tulio Ribeiro    Yi Zhang

Microsoft Research

2023

Artificial intelligence (AI) researchers have been developing and refining large language models (LLMs) that exhibit remarkable capabilities across a variety of domains and tasks, challenging our understanding of learning and cognition. The latest model developed by OpenAI, GPT-4 [Ope23], was trained using an unprecedented scale of compute and data. In this paper, we report on our investigation of an early version of GPT-4, when it was still in active development by OpenAI. We contend that (this early version of) GPT-4 is part of a new cohort of LLMs (along with ChatGPT and Google’s PaLM for example) that exhibit more general intelligence than previous AI models. We discuss the rising capabilities and implications of these models. We demonstrate that, beyond its mastery of language, GPT-4 can solve novel and difficult tasks that span mathematics, coding, vision, medicine, law, psychology and more, without needing any special prompting. Moreover, in all of these tasks, GPT-4’s performance is strikingly close to human-level performance, and often vastly surpasses prior models such as ChatGPT. Given the breadth and depth of GPT-4’s capabilities, we believe that it could reasonably be viewed as an early (yet still incomplete) version of an artificial general intelligence (AGI) system. In our exploration of GPT-4, we put special emphasis on discovering its limitations, and we discuss the challenges ahead for advancing towards deeper and more comprehensive versions of AGI, including the possible need for pursuing a new paradigm that moves beyond next-word prediction. We conclude with reflections on societal influences of the recent technological leap and future research directions.

Previously on CENG501

# Limits of LLMs

n 2023

---

## Faith and Fate: Limits of Transformers on Compositionality

---

Nouha Dziri<sup>1\*</sup>, Ximing Lu<sup>1,2\*</sup>, Melanie Sclar<sup>2\*</sup>, Xiang Lorraine Li<sup>1†</sup>, Liwei Jiang<sup>1,2 †</sup>,  
Bill Yuchen Lin<sup>1</sup>, Peter West<sup>1,2</sup>, Chandra Bhagavatula<sup>1</sup>, Ronan Le Bras<sup>1</sup>, Jena D. Hwang<sup>1</sup>,  
Soumya Sanyal<sup>3</sup>, Sean Welleck<sup>1,2</sup>, Xiang Ren<sup>1,3</sup>, Allyson Ettinger<sup>1,4</sup>,  
Zaid Harchaoui<sup>1,2</sup>, Yejin Choi<sup>1,2</sup>

<sup>1</sup>Allen Institute for Artificial Intelligence    <sup>2</sup>University of Washington

<sup>3</sup>University of Southern California    <sup>4</sup>University of Chicago

[nouhad@allenai.org](mailto:nouhad@allenai.org), [ximinglu@allenai.org](mailto:ximinglu@allenai.org), [msclar@cs.washington.edu](mailto:msclar@cs.washington.edu)

Transformer large language models (LLMs) have sparked admiration for their exceptional performance on tasks that demand intricate multi-step reasoning. Yet, these models simultaneously show failures on surprisingly trivial problems. This begs the question: Are these errors incidental, or do they signal more substantial limitations? In an attempt to demystify Transformers, we investigate the limits of these models across three representative *compositional* tasks—multi-digit multiplication, logic grid puzzles, and a classic dynamic programming problem. These tasks require breaking problems down into sub-steps and synthesizing these steps into a precise answer. We formulate compositional tasks as computation graphs to systematically quantify the level of complexity, and break down reasoning steps into intermediate sub-procedures. Our empirical findings suggest that Transformers solve compositional tasks by reducing multi-step compositional reasoning into linearized subgraph matching, without necessarily developing systematic problem-solving skills. To round off our empirical study, we provide theoretical arguments on abstract multi-step reasoning problems that highlight how Transformers’ performance will rapidly decay with increased task complexity.

Previously on CENG501

# Risks of LLMs

- Risk area 1: Discrimination, Hate speech and Exclusion
  - Social stereotypes and unfair discrimination
  - Hate speech and offensive language
  - Exclusionary norms
  - Lower performance for some languages and social groups

## Taxonomy of Risks posed by Language Models

Laura Weidinger* DeepMind UK	Jonathan Uesato DeepMind UK	Maribeth Rauh DeepMind UK	Conor Griffin DeepMind UK
Po-Sen Huang DeepMind UK	John Mellor DeepMind UK	Amelia Glaese DeepMind UK	Myra Cheng† DeepMind UK
Borja Balle DeepMind UK	Atoosa Kasirzadeh‡ DeepMind UK	Courtney Biles DeepMind UK	Sasha Brown DeepMind UK
Zac Kenton DeepMind UK	Will Hawkins DeepMind UK	Tom Stepleton DeepMind UK	Abeba Birhane§ DeepMind UK
Lisa Anne Hendricks DeepMind UK	Laura Rimell DeepMind UK	William Isaac DeepMind UK	Julia Haas DeepMind UK
	Sean Legassick DeepMind UK	Geoffrey Irving DeepMind UK	Iason Gabriel DeepMind UK

2022

Previously on CENG501

## On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?

Emily M. Bender\*  
ebender@uw.edu  
University of Washington  
Seattle, WA, USA

Angelina McMillan-Major  
aymm@uw.edu  
University of Washington  
Seattle, WA, USA

Timnit Gebru\*  
timnit@blackinai.org  
Black in AI  
Palo Alto, CA, USA

Shmargaret Shmitchell  
shmargaret.shmitchell@gmail.com  
The Aether

FaccT2021

- Environmental & financial costs
- Require vast data
  - Not necessarily diverse
  - Includes bias
  - Accountability/liability
- Stochastic Parrots

Year	Model	# of Parameters	Dataset Size
2019	BERT [39]	3.4E+08	16GB
2019	DistilBERT [113]	6.60E+07	16GB
2019	ALBERT [70]	2.23E+08	16GB
2019	XLNet (Large) [150]	3.40E+08	126GB
2020	ERNIE-GEN (Large) [145]	3.40E+08	16GB
2019	RoBERTa (Large) [74]	3.55E+08	161GB
2019	MegatronLM [122]	8.30E+09	174GB
2020	T5-11B [107]	1.10E+10	745GB
2020	T-NLG [112]	1.70E+10	174GB
2020	GPT-3 [25]	1.75E+11	570GB
2020	GShard [73]	6.00E+11	–
2021	Switch-C [43]	1.57E+12	745GB

Table 1: Overview of recent large language models

# Today

- Using Pretrained LLMs
  - In-context Learning
  - Retrieval Augmented Generation
  - Finetuning



# Administrative Notes

- No quiz this week
- Time plan for the projects
  1. **Milestone (November 24, midnight):**
    - Github repo will be ready
    - Read & understand the paper
    - Download the datasets
    - Prepare the Readme file excluding the results & conclusion
  2. **Milestone (December 8, midnight)**
    - The results of the first experiment
  3. **Milestone (January 5, midnight)**
    - Final report (Readme file)
    - Repo with all code & trained models

# In-Context Learning (Prompt Engineering)

# In-context Learning (Prompt Engineering)

## Three ways of in-context learning:

### Zero-shot

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

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

### One-shot

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

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

In a single sequence input, the prompted example can learn from previous demonstrations.

### Few-shot

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

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

# In-context Learning

- Simplifies providing examples (from human experts)
  - As opposed to updating the weights of the network via finetuning
- “Example-based Specification”, Programming by example.
- Provides on par performance with supervised models
- More params, better in-context learning

Circulation revenue has increased by 5% in Finland. // Finance

They defeated ... in the NFC Championship Game. // Sports

Apple ... development of in-house chips. // Tech

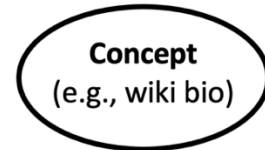
The company anticipated its operating profit to improve. // \_\_\_\_\_



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

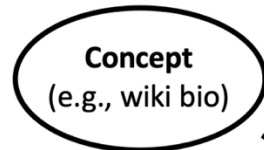
# In-context Learning: How/Why Does it Work?

1. **Pretraining documents** are conditioned on a **latent concept** (e.g., biographical text)



Albert Einstein was a German theoretical physicist, widely acknowledged to be one of the greatest physicists of all time. Einstein is best known for developing the theory of relativity, but he also ....

2. Create **independent examples** from a **shared concept**. If we focus on full names, wiki bios tend to relate them to nationalities.



Input (x)	Output (y)	Delimiter
Albert Einstein was	German	\n
Mahatma Gandhi was	Indian	\n
Marie Curie was	?	...brilliant? ...Polish?

3. **Concatenate examples into a prompt** and predict next word(s). **Language model (LM)** implicitly infers the **shared concept** across examples despite the unnatural concatenation



Xie et al., "An Explanation of In-context Learning as Implicit Bayesian Inference", ICLR 2022.

# In-context Learning: How/Why Does it Work?

A Bayesian interpretation:

$$p(\text{output}|\text{prompt}) = \int_{\text{concept}} p(\text{output}|\text{concept}, \text{prompt})p(\text{concept}|\text{prompt})d(\text{concept})$$

- During pretraining, the network learns a latent concept space.
- With the prompt, we provide sufficient examples to estimate the most relevant concept –  $p(\text{concept} | \text{prompt})$ .

Xie et al., “An Explanation of In-context Learning as Implicit Bayesian Inference”, ICLR 2022.

# In-context Learning: How/Why Does it Work?

Circulation revenue has increased by 5% in Finland. // Finance

They defeated ... in the NFC Championship Game. // Sports

Apple ... development of in-house chips. // Tech

**Not an easy task since examples might be unrelated to each other!**

*“We can think of the training examples as providing a signal for Bayesian inference. In particular, the transitions within training examples (green in the figure above) allow the LM to infer the latent concept they all share. In a prompt, the green transitions come from the input distribution (the transitions inside the news sentence), the output distribution (the topic word), the format (syntax of news sentence), and the input-output mapping (relation between the news and the topic) all provide signal for Bayesian inference.”*

# In-context learning: Task vectors

Hendel et al., "In-Context Learning Creates Task Vectors", 2023.

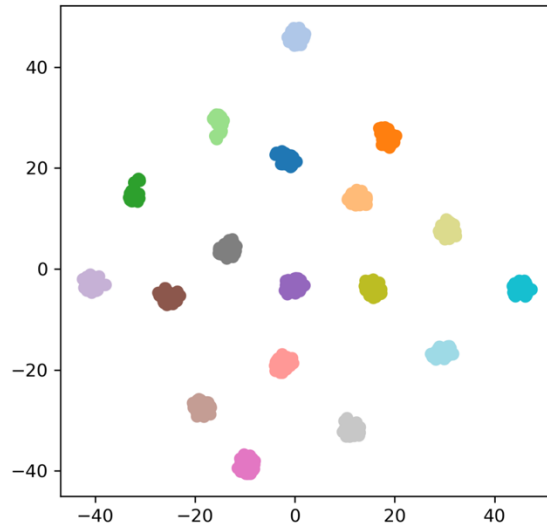


Figure 5: **A t-SNE plot of task vectors.** A 2D t-SNE plot visualizing 50 task vectors for each task, each generated from a different choice of  $S$  and  $x'$  using LLaMA 7B. Points are color-coded according to the task. Each task can be seen to form its own distinct cluster.

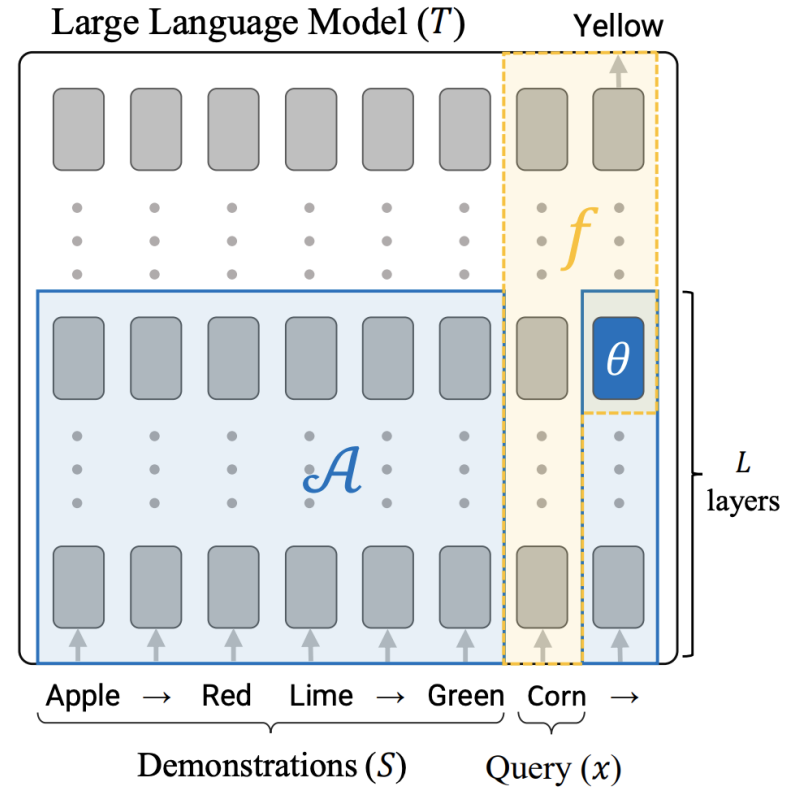


Figure 1: **ICL as learning in a Hypothesis Class.** In ICL, one provides an LLM with a prompt including demonstrations  $S$  of some task, and a query  $x$ . The model generates the output for  $x$  (here "Yellow"). We show that the underlying process can be broken down into two parts:  $\mathcal{A}$ , a "learning algorithm" (marked in blue), computes a query-agnostic vector  $\theta(S)$ , which we view as a parameter of a function in a hypothesis class. The second part, denoted by  $f$  and marked in yellow, is the application of the rule defined by  $\theta$  on the query  $x$ , without direct dependence on  $S$ .



# In-context Learning: Important Factors

Min et al., “Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?”, 2022.

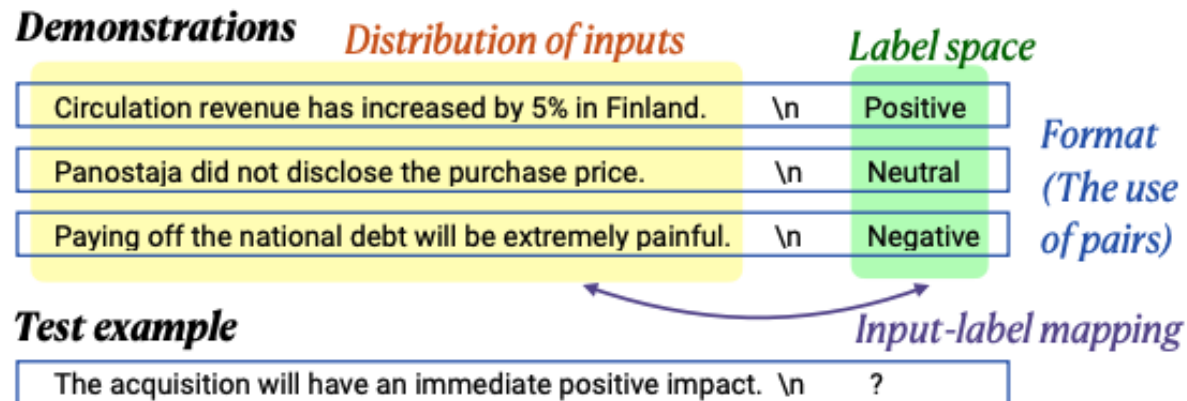


Figure 7: Four different aspects in the demonstrations: the input-label mapping, the distribution of the input text, the label space, and the use of input-label pairing as the format of the demonstrations.

## Abstract

Large language models (LMs) are able to in-context learn—perform a new task via inference alone by conditioning on a few input-label pairs (demonstrations) and making predictions for new inputs. However, there has been little understanding of *how* the model learns and *which* aspects of the demonstrations contribute to end task performance. In this paper, we show that ground truth demonstrations are in fact not required—randomly replacing labels in the demonstrations barely hurts performance on a range of classification and multi-choice tasks, consistently over 12 different models including GPT-3. Instead, we find that other aspects of the demonstrations are the key drivers of end task performance, including the fact that they provide a few examples of (1) the label space, (2) the distribution of the input text, and (3) the overall format of the sequence. Together, our analysis provides a new way of understanding how and why in-context learning works, while opening up new questions about how much can be learned from large language models through inference alone.

# In-context Learning: Important Factors

Min et al., “Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?”, 2022.

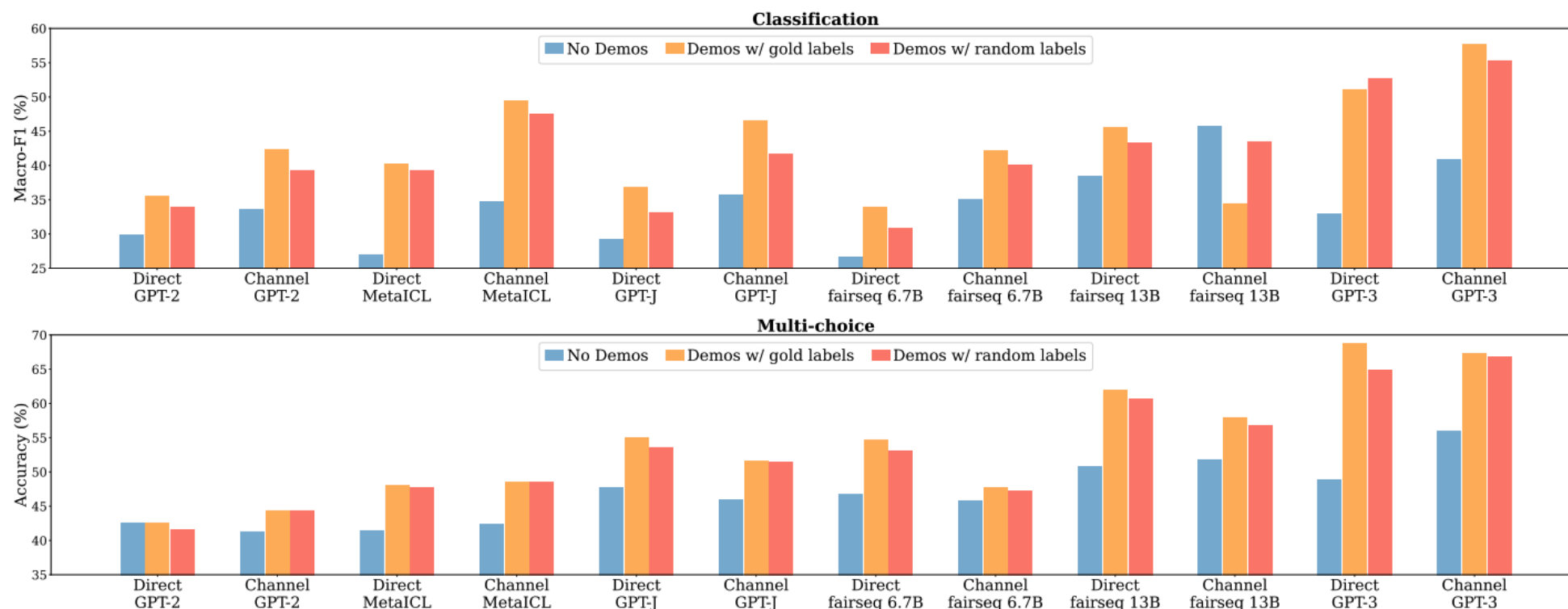


Figure 3: Results when using no-demonstrations, demonstrations with gold labels, and demonstrations with random labels in classification (top) and multi-choice tasks (bottom). The first eight models are evaluated on 16 classification and 10 multi-choice datasets, and the last four models are evaluated on 3 classification and 3 multi-choice datasets. See Figure 11 for numbers comparable across all models. **Model performance with random labels is very close to performance with gold labels** (more discussion in Section 4.1).

# In-context Learning: Important Factors

Min et al., “Rethinking the Role of Demonstrations:  
What Makes In-Context Learning Work?”, 2022.

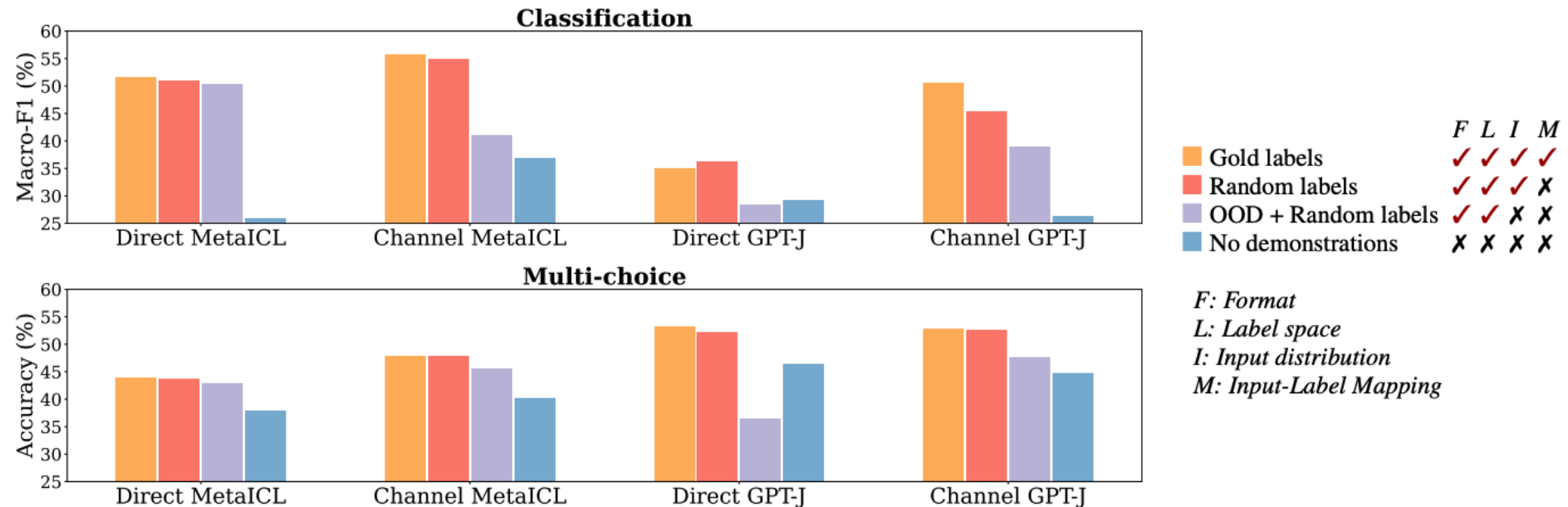


Figure 8: Impact of the distribution of the inputs. Evaluated in classification (top) and multi-choice (bottom). The impact of the distribution of the input text can be measured by comparing ■ and ■. The gap is substantial, with an exception in Direct MetaICL (discussion in Section 5.1).

# In-context Learning: Important Factors

Min et al., “Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?”, 2022.

They conclude that

- (1) “the label space and the distribution of the input text specified by the demonstrations are both key to in-context learning (regardless of whether the labels are correct for individual inputs)”
- (2) “specifying the overall format is also crucial, e.g., when the label space is unknown, using random English words as labels is significantly better than using no labels”
- (3) “meta-training with an in-context learning objective (Min et al., 2021b) magnifies these effects—the models almost exclusively exploit simpler aspects of the demonstrations like the format rather than the input-label mapping.”

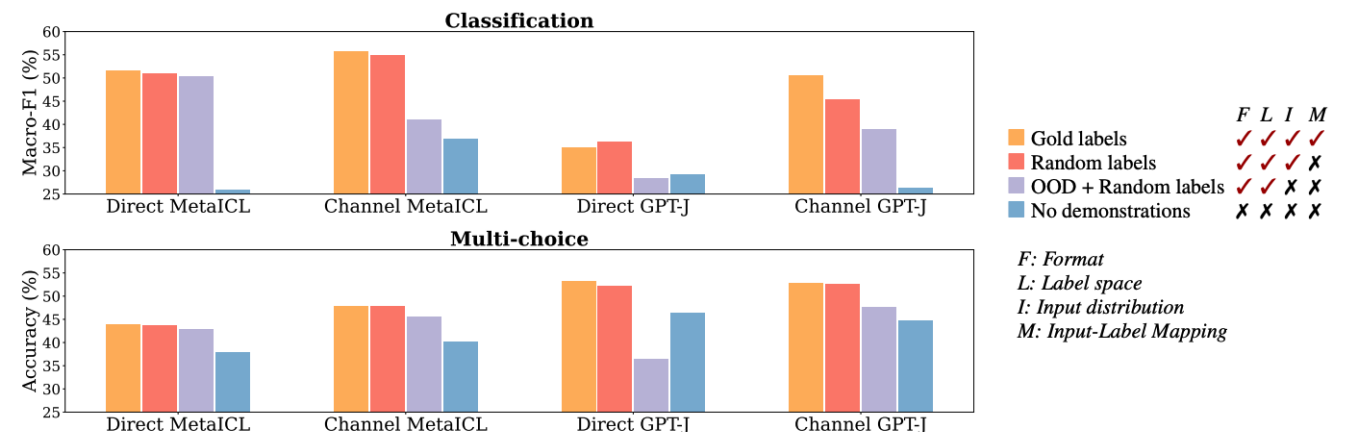
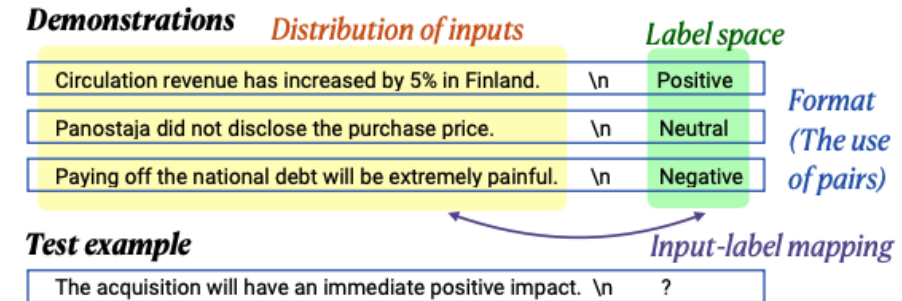


Figure 8: Impact of the distribution of the inputs. Evaluated in classification (top) and multi-choice (bottom). The impact of the distribution of the input text can be measured by comparing ■ and ■. The gap is substantial, with an exception in Direct MetaICL (discussion in Section 5.1).

# In-context Learning: Important Factors

In-context Learning (ICL) can be unstable:

- Min et al. (<https://arxiv.org/abs/2202.12837>):
  - “LLMs rely strongly on superficial cues. ICL acts more as a pattern recognition procedure, than as an actual “learning” procedure: the input-output mappings that are provided allow a model to retrieve similar examples it has been exposed to during training, but the moment you start flipping labels or the template model performance breaks.”
- Weber et al. (<https://arxiv.org/abs/2310.13486>):
  - “Previous research has also shown that ICL is highly unstable. For example, the order of in-context examples (Lu et al., 2022), the recency of certain labels in the context (Zhao et al., 2021) or the format of the prompt (Mishra et al., 2022) as well as the distribution of training examples and the label space (Min et al., 2022) strongly influence model performance. Curiously, whether the labels provided in the examples are correct is less important (Min et al., 2022). However, these findings are not uncontested: Yoo et al. (2022) paint a more differentiated picture, demonstrating that in-context input-label mapping does matter, but that it depends on other factors such as model size or instruction verbosity. Along a similar vein, Wei et al. (2023) show that in-context learners can acquire new semantically non-sensical mappings from in-context examples if presented in a specific setup.”

# In-context Learning: Limitations

- **Model Parameters and Scale:** Scale helps!
- **Training Data Dependency:** Dataset size matters!
- **Domain Specificity:**
  - "While LLMs can generalize across various tasks, there might be limitations when dealing with highly specialized domains. Domain-specific data might be required to achieve optimal results."
- **Model Fine-Tuning:**
  - "Even with ICL, there might be scenarios where model fine-tuning becomes necessary to cater to specific tasks or correct undesirable emergent abilities."
- **Ethics and Fairness**
- **Privacy and Security**

# Chain of Thought

## Standard Prompting

### Model Input

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

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The answer is 27. ❌

## Chain-of-Thought Prompting

### Model Input

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

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✅

# Zero-shot Chain of Thought

(a) Few-shot

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

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. **X**

(b) Few-shot-CoT

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

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4. **✓**

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 **X**

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. **✓**



# Meta Prompting

Standard  
Prompting

Problem: Find the domain of the expression  $\frac{\sqrt{x-2}}{\sqrt{5-x}}$ .

Solution: The expressions inside each square root must be non-negative. Therefore,  $x - 2 \geq 0$ , so  $x \geq 2$ , and  $5 - x \geq 0$ , so  $x \leq 5$ . Also, the denominator cannot be equal to zero, so  $5 - x > 0$ , which gives  $x < 5$ . Therefore, the domain of the expression is  $[2, 5)$ . Final Answer: The final answer is  $[2, 5)$ . I hope it is correct.

Problem: If  $\det \mathbf{A} = 2$  and  $\det \mathbf{B} = 12$ , then find  $\det(\mathbf{AB})$ .

Solution: We have that  $\det(\mathbf{AB}) = (\det \mathbf{A})(\det \mathbf{B}) = (2)(12) = 24$ . Final Answer: The final answer is 24. I hope it is correct.

...

Meta  
Prompting

**Problem Statement:**

- **Problem:** [question to be answered]

**Solution Structure:**

1. Begin the response with "Let's think step by step."
2. Follow with the reasoning steps, ensuring the solution process is broken down clearly and logically.
3. End the solution with the final answer encapsulated in a LaTeX-formatted box,  $\boxed{\dots}$ , for clarity and emphasis.
4. Finally, state "The answer is [final answer to the problem].", with the final answer presented in LaTeX notation.

Figure 1: A structure meta prompt presented in markdown format for solving MATH [17] problems.

# Chain of Symbol

Hu et al., “Chain-of-Symbol Prompting Elicits Planning in Large Language Models”, 2023.

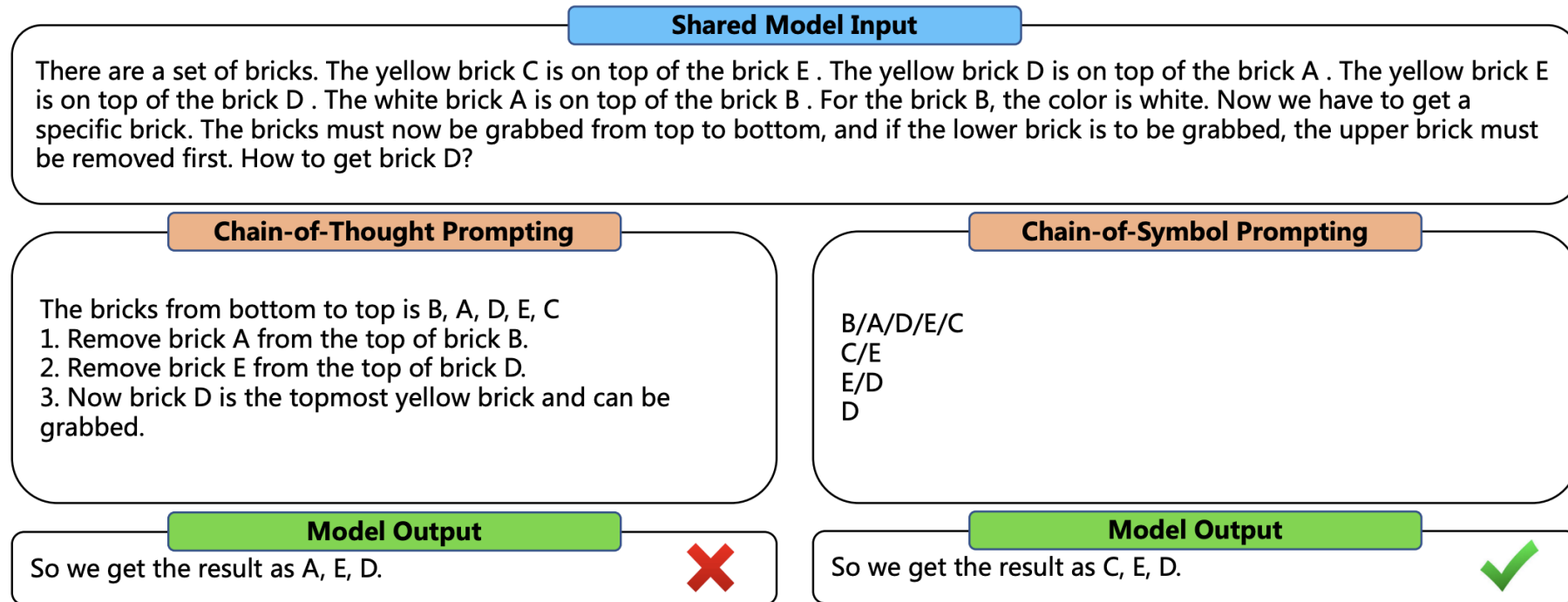


Figure 1: An example for comparison between Chain-of-Thought (CoT) and Chain-of-Symbol (COS) that elicits large language models in tackling complex planning tasks with higher performance and fewer input tokens. We let the model generate CoT/COS during inference in a few-shot manner. Results were taken in May 2023 with ChatGPT and can be subject to change.

# Chain of Symbol

Hu et al., “Chain-of-Symbol Prompting Elicits Planning in Large Language Models”, 2023.

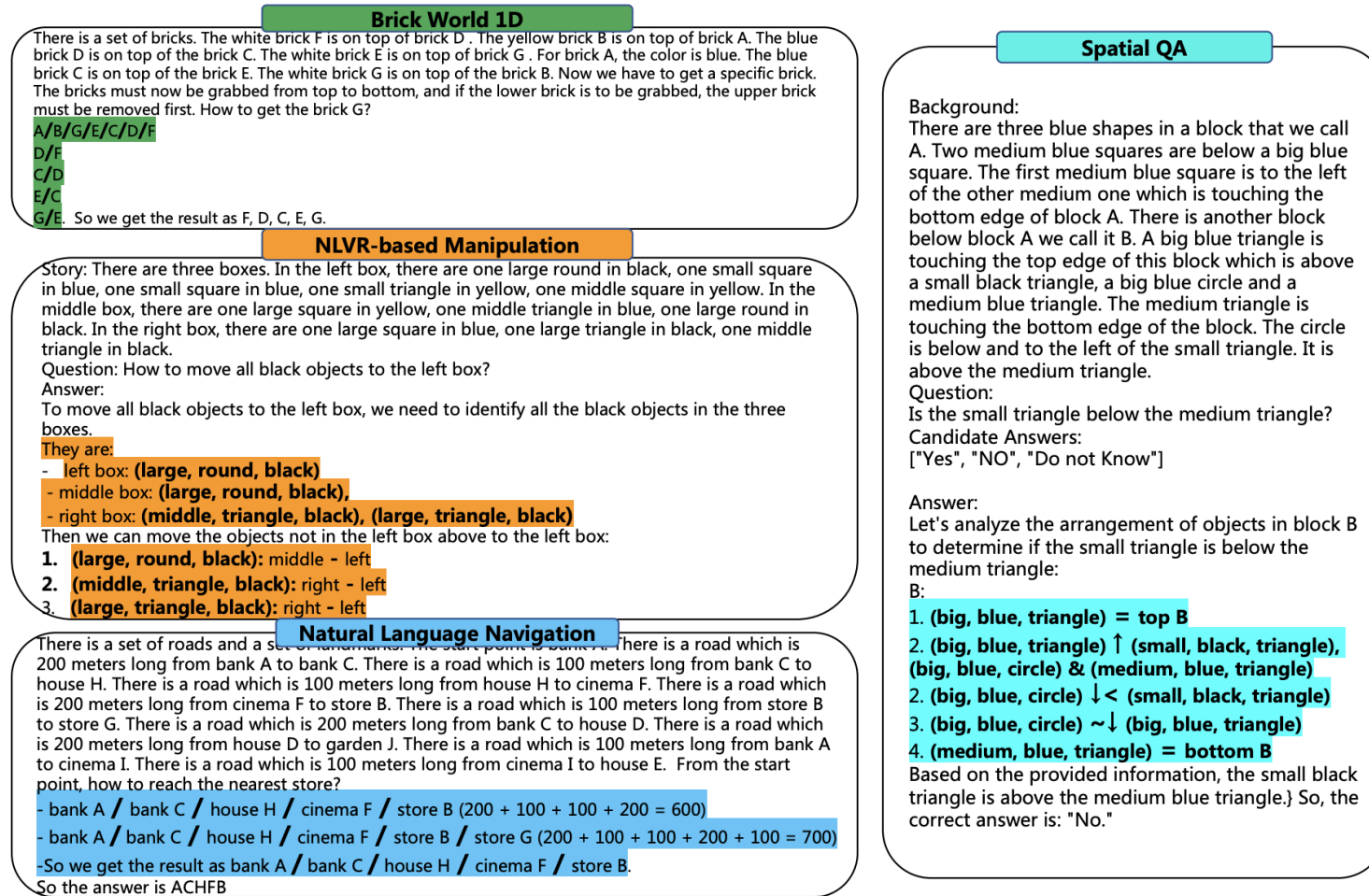


Figure 2: <input, Chain of Symbol, output> example triples for our three proposed tasks: Brick World, NLVR-based Manipulation, and Natural Language Navigation, and SPARTUN dataset (Mirzaee & Kordjamshidi, 2022). Chains of Symbols are highlighted.

# Generated knowledge prompting

It remains an open question whether incorporating external knowledge benefits commonsense reasoning while maintaining the flexibility of pretrained sequence models. To investigate this question, we develop generated knowledge prompting, which consists of **generating knowledge from a language model, then providing the knowledge as additional input when answering a question.** Our method does not require task-specific supervision for knowledge integration, or access to a structured knowledge base, yet it improves performance of large-scale, state-of-the-art models on four commonsense reasoning tasks, achieving state-of-the-art results on numerical commonsense (NumerSense), general commonsense (CommonsenseQA 2.0), and scientific commonsense (QASC) benchmarks. Generated knowledge prompting highlights large-scale language models as flexible sources of external knowledge for improving common-

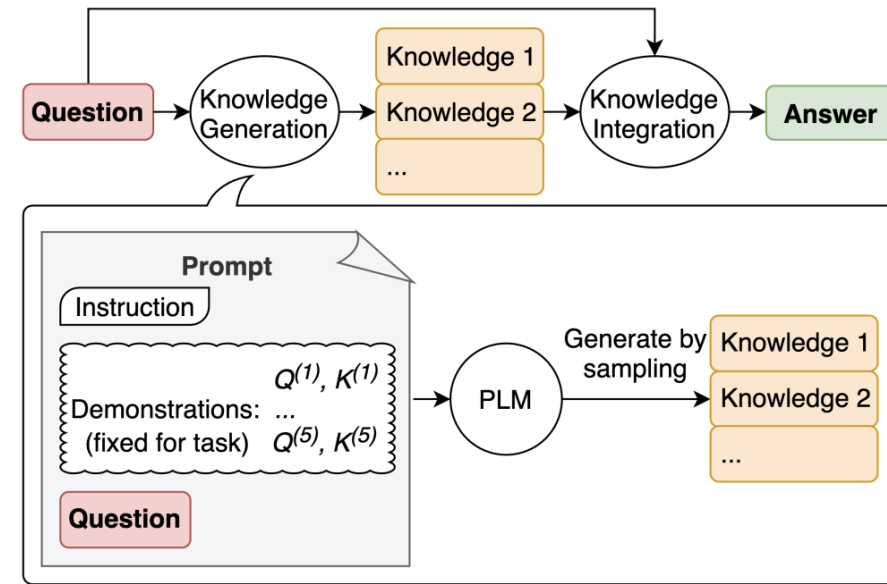


Figure 1: Generated knowledge prompting involves (i) using few-shot demonstrations to generate question-related knowledge statements from a language model; (ii) using **a second language model** to make predictions with each knowledge statement, then selecting the highest-confidence prediction.

# Generated knowledge prompting

Input: Greece is larger than Mexico.

Knowledge: Greece is approximately 131,957 sq km, while Mexico is approximately 1,964,375 sq km, making Mexico 1,389% larger than Greece.

Input: Glasses always fog up.

Knowledge: Condensation occurs on eyeglass lenses when water vapor from your sweat, breath, and ambient humidity lands on a cold surface, cools, and then changes into tiny drops of liquid, forming a film that you see as fog. Your lenses will be relatively cool compared to your breath, especially when the outside air is cold.

Input: A fish is capable of thinking.

Knowledge: Fish are more intelligent than they appear. In many areas, such as memory, their cognitive powers match or exceed those of 'higher' vertebrates including non-human primates. Fish's long-term memories help them keep track of complex social relationships.

Input: A common effect of smoking lots of cigarettes in one's lifetime is a higher than normal chance of getting lung cancer.

Knowledge: Those who consistently averaged less than one cigarette per day over their lifetime had nine times the risk of dying from lung cancer than never smokers. Among people who smoked between one and 10 cigarettes per day, the risk of dying from lung cancer was nearly 12 times higher than that of never smokers.

Input: A rock is the same size as a pebble.

Knowledge: A pebble is a clast of rock with a particle size of 4 to 64 millimetres based on the Udden-Wentworth scale of sedimentology. Pebbles are generally considered larger than granules (2 to 4 millimetres diameter) and smaller than cobbles (64 to 256 millimetres diameter).

Input: Part of golf is trying to get a higher point total than others.

Knowledge:

# Self-consistency

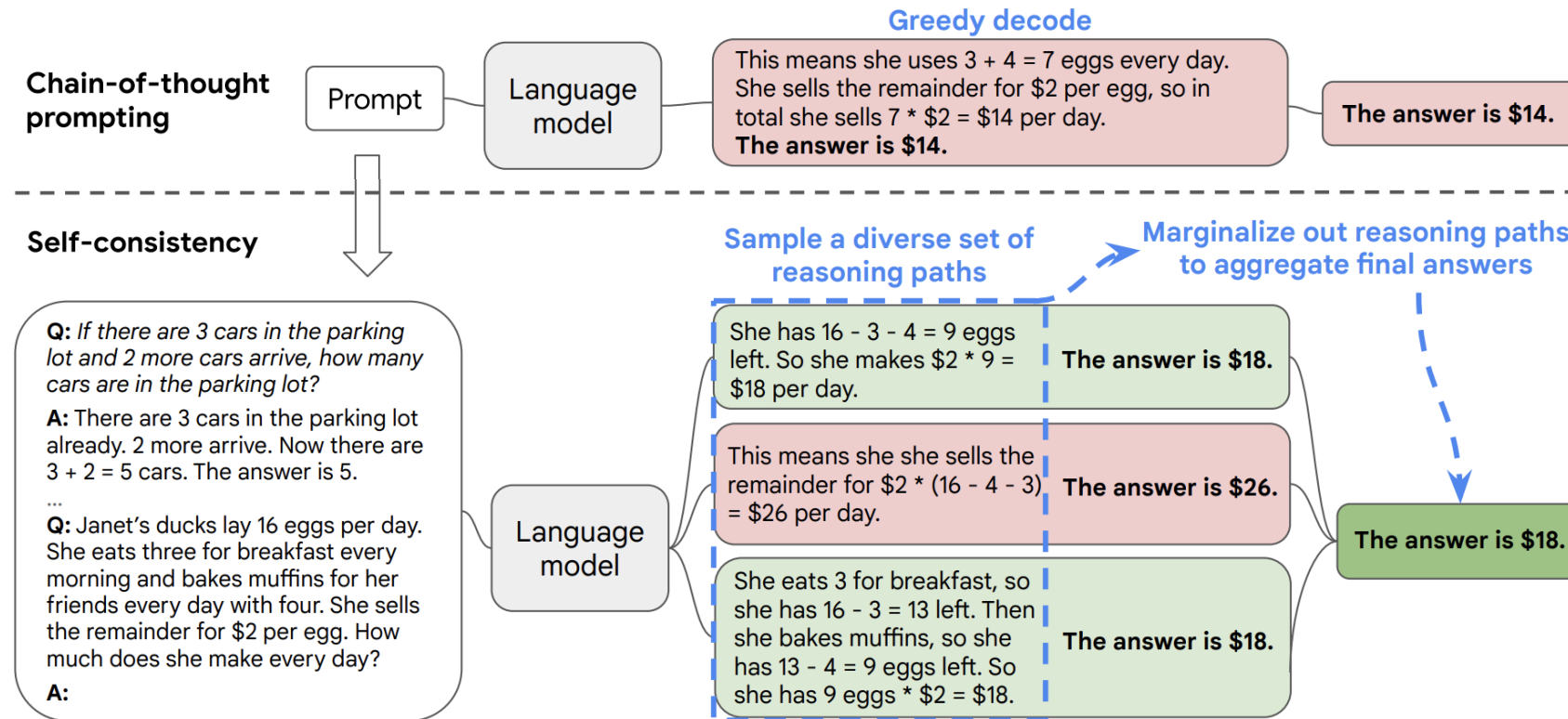
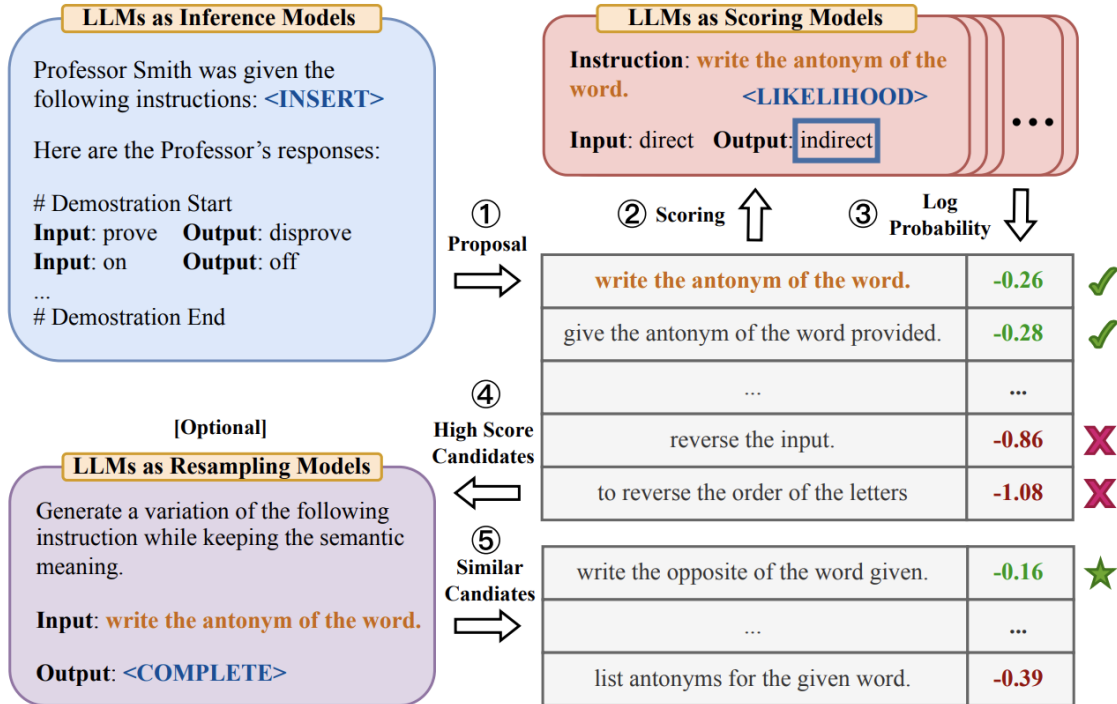


Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the “greedy decode” in CoT prompting by sampling from the language model’s decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

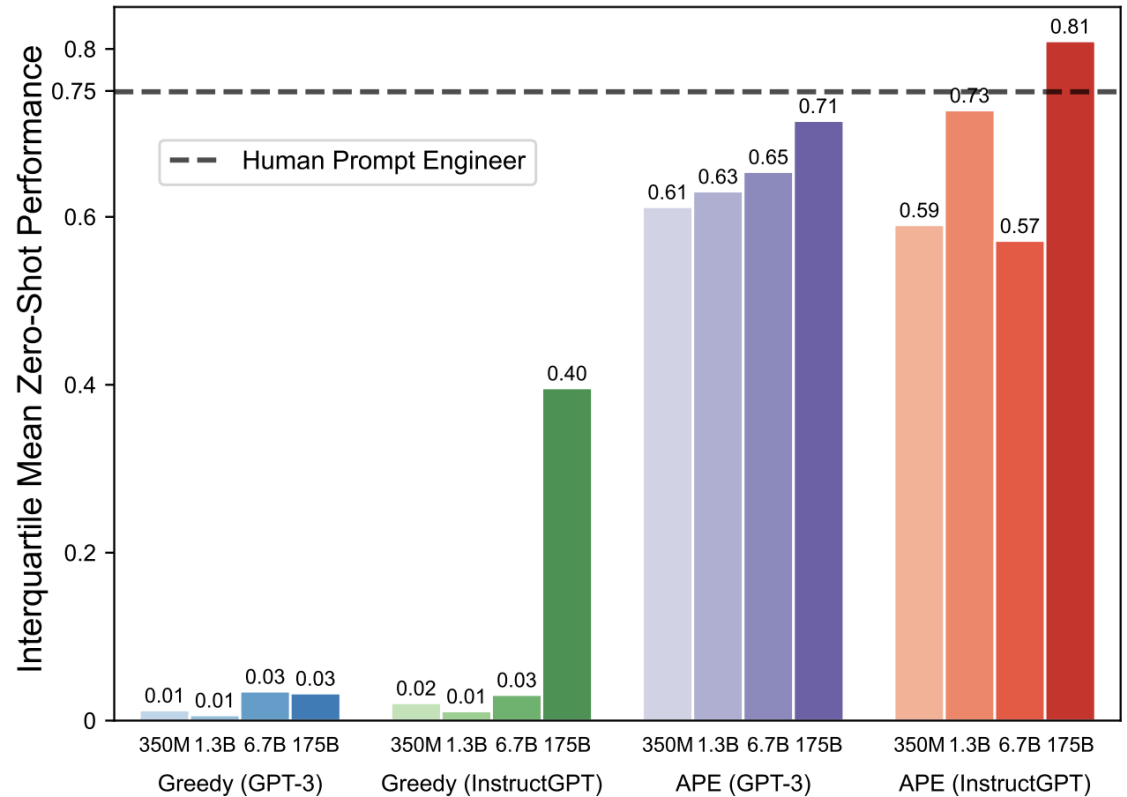
# Automatic Prompt Engineer

Zhou et al., "LARGE LANGUAGE MODELS ARE HUMAN-LEVEL PROMPT ENGINEERS", 2023.

✓ Keep the high score candidates    ✗ Discard the low score candidates    ★ Final selected prompt with highest score



(a) Automatic Prompt Engineer (APE) workflow



(b) Interquartile mean across 24 tasks

# Tree of Thoughts Prompting

Yao et al., "Tree of Thoughts: Deliberate Problem Solving with Large Language Models", 2023.

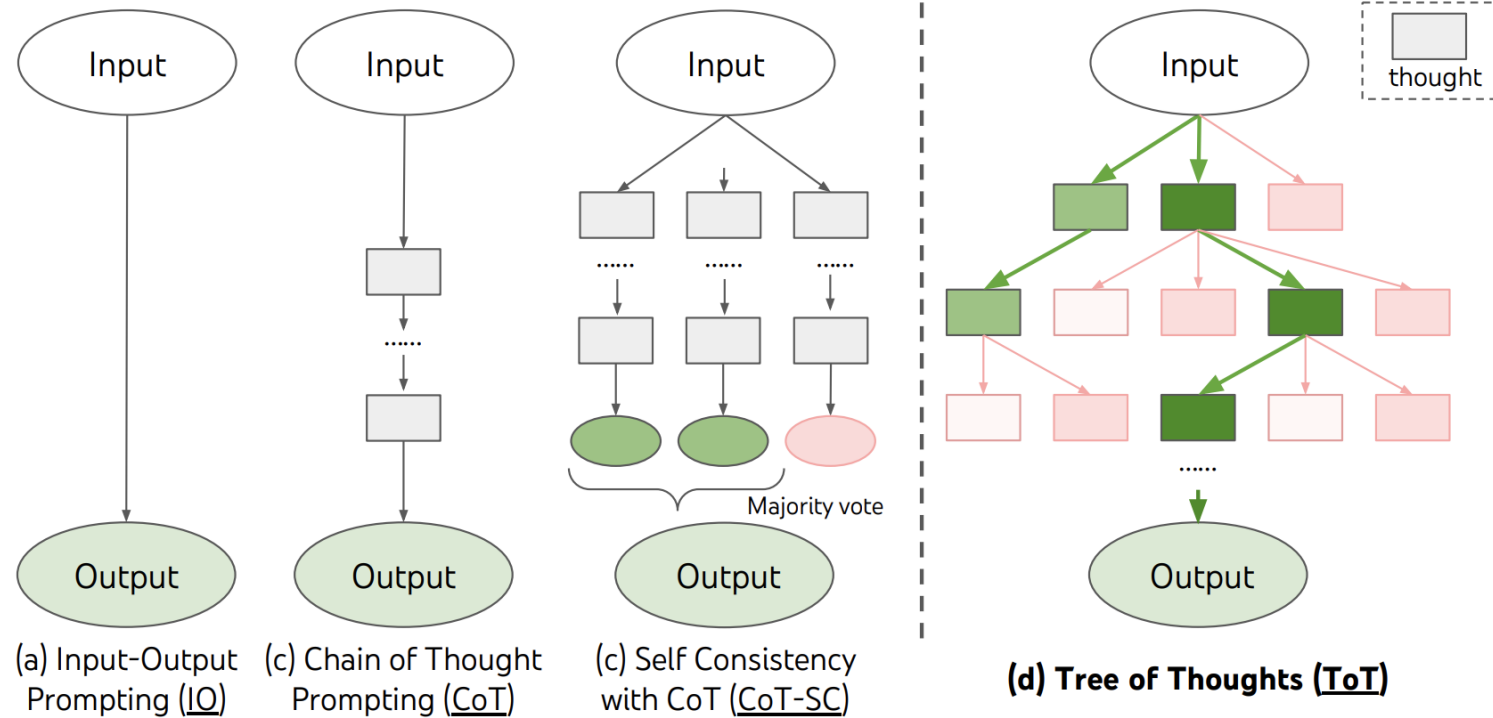


Figure 1: Schematic illustrating various approaches to problem solving with LLMs. Each rectangle box represents a *thought*, which is a coherent language sequence that serves as an intermediate step toward problem solving. See concrete examples of how thoughts are generated, evaluated, and searched in Figures 2,4,6.



# Tree of Thoughts Prompting

---

**Algorithm 1** ToT-BFS( $x, p_\theta, G, k, V, T, b$ )

**Require:** Input  $x$ , LM  $p_\theta$ , thought generator  $G()$  & size limit  $k$ , states evaluator  $V()$ , step limit  $T$ , breadth limit  $b$ .

$S_0 \leftarrow \{x\}$

**for**  $t = 1, \dots, T$  **do**

$S'_t \leftarrow \{[s, z] \mid s \in S_{t-1}, z_t \in G(p_\theta, s, k)\}$

$V_t \leftarrow V(p_\theta, S'_t)$

$S_t \leftarrow \arg \max_{S \subset S'_t, |S|=b} \sum_{s \in S} V_t(s)$

**end for**

**return**  $G(p_\theta, \arg \max_{s \in S_T} V_T(s), 1)$

---

---

**Algorithm 2** ToT-DFS( $s, t, p_\theta, G, k, V, T, v_{th}$ )

**Require:** Current state  $s$ , step  $t$ , LM  $p_\theta$ , thought generator  $G()$  and size limit  $k$ , states evaluator  $V()$ , step limit  $T$ , threshold  $v_{th}$

**if**  $t > T$  **then** record output  $G(p_\theta, s, 1)$

**end if**

**for**  $s' \in G(p_\theta, s, k)$  **do**   ▷ sorted candidates

**if**  $V(p_\theta, \{s'\})(s) > v_{thres}$  **then** ▷ pruning

        DFS( $s', t + 1$ )

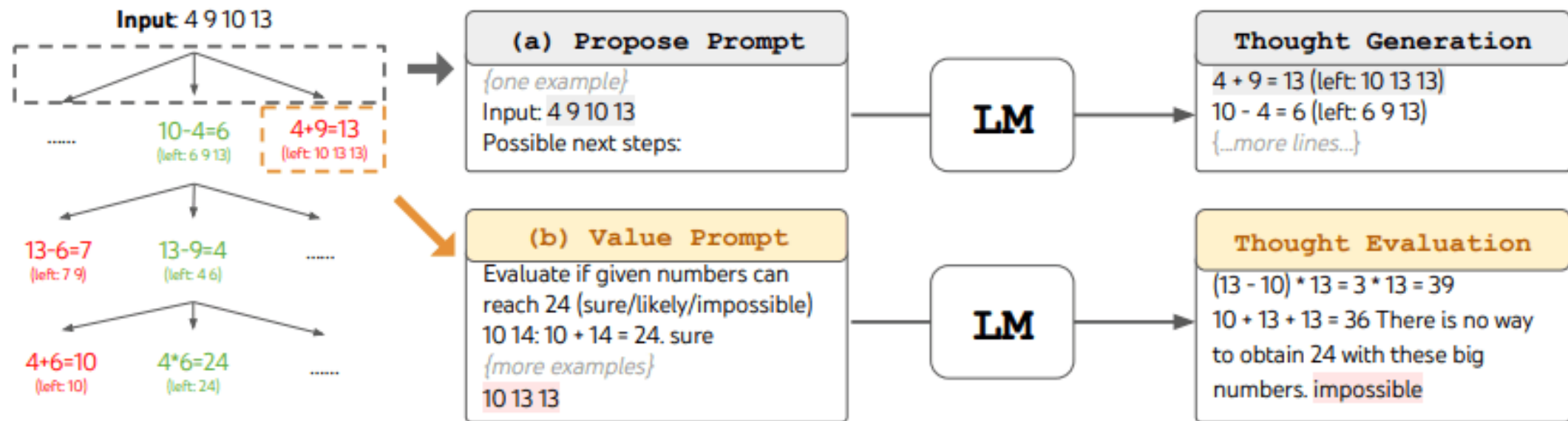
**end if**

**end for**

---

# Tree of Thoughts Prompting

Yao et al., "Tree of Thoughts: Deliberate Problem Solving with Large Language Models", 2023.



# Tree of Thoughts Prompting

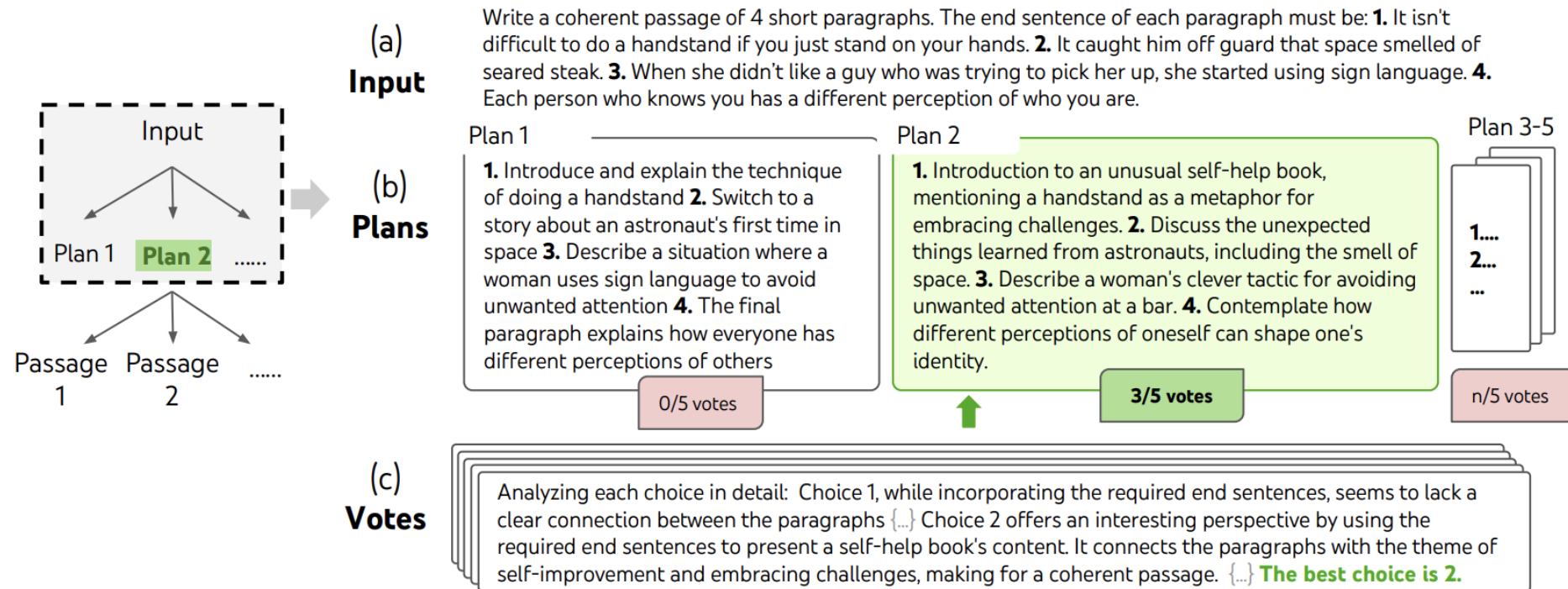


Figure 4: A step of deliberate search in a randomly picked Creative Writing task. Given the input, the LM samples 5 different plans, then votes 5 times to decide which plan is best. The majority choice is used to consequently write the output passage with the same sample-vote procedure.

# Program-Aided LMs

Gao et al., "PAL: Program-aided Language Models", 2023.

## Chain-of-Thought (Wei et al., 2022)

Input

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

A: Roger started with 5 tennis balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold  $93 + 39 = 132$  loaves. The grocery store returned 6 loaves. So they had  $200 - 132 - 6 = 62$  loaves left.

The answer is 62.



## Program-aided Language models (this work)

Input

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

A: Roger started with 5 tennis balls.

```
tennis_balls = 5
```

```
2 cans of 3 tennis balls each is
```

```
bought_balls = 2 * 3
```

```
tennis_balls
```

```
answer = tennis_balls + bought_balls
```

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves

```
loaves_baked = 200
```

```
They sold 93 in the morning and 39 in the afternoon
```

```
loaves_sold_morning = 93
```

```
loaves_sold_afternoon = 39
```

```
The grocery store returned 6 loaves.
```

```
loaves_returned = 6
```

The answer is

```
answer = loaves_baked - loaves_sold_morning
```

```
- loaves_sold_afternoon + loaves_returned
```

```
>>> print(answer)
```

```
74
```



# Other Types of Prompting

- Least-to-most prompting
- Complexity-based prompting
- Self-refine
- Maieutic prompting
- Using gradient descent to search for prompts: "prefix-tuning", "prompt tuning" or "soft prompting"
- Prompt injection

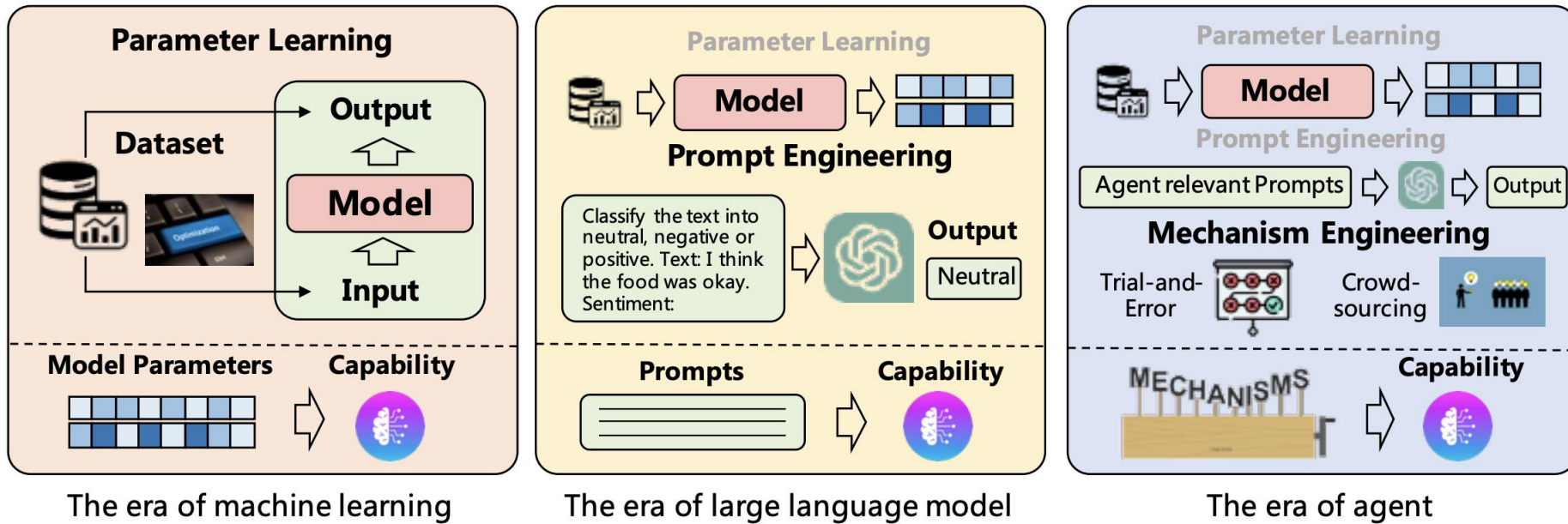
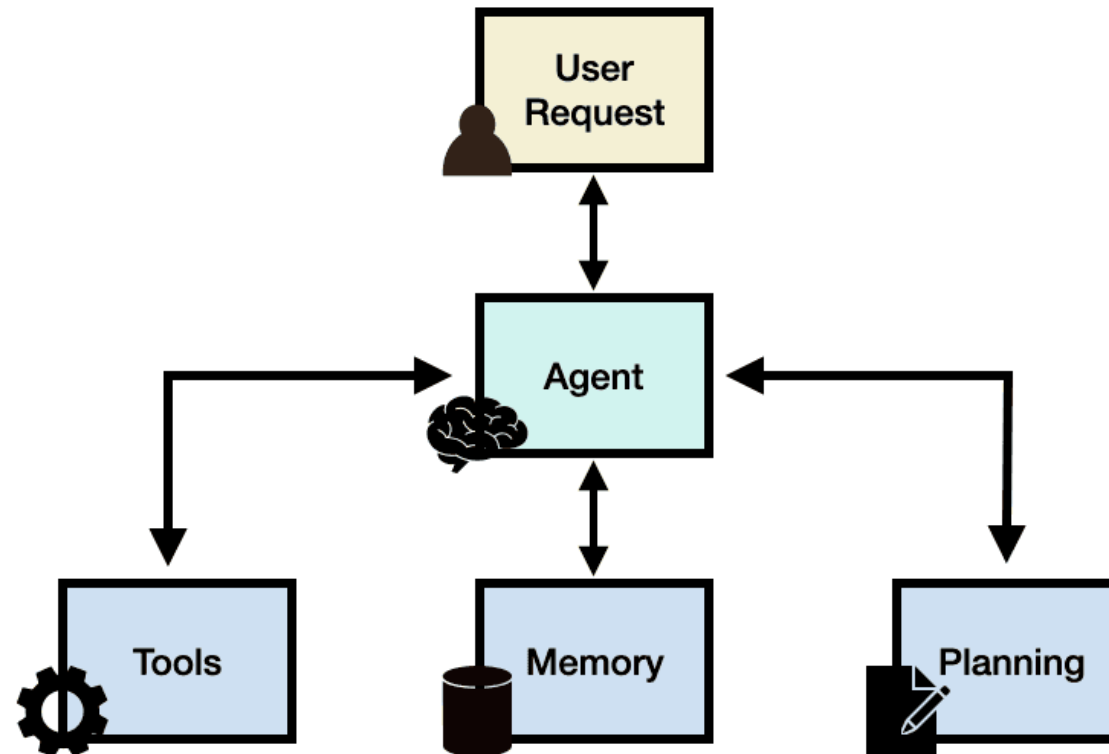


Figure: Wang et al., "A Survey on Large Language Model based Autonomous Agents", 2024.

# LLMs as Agents

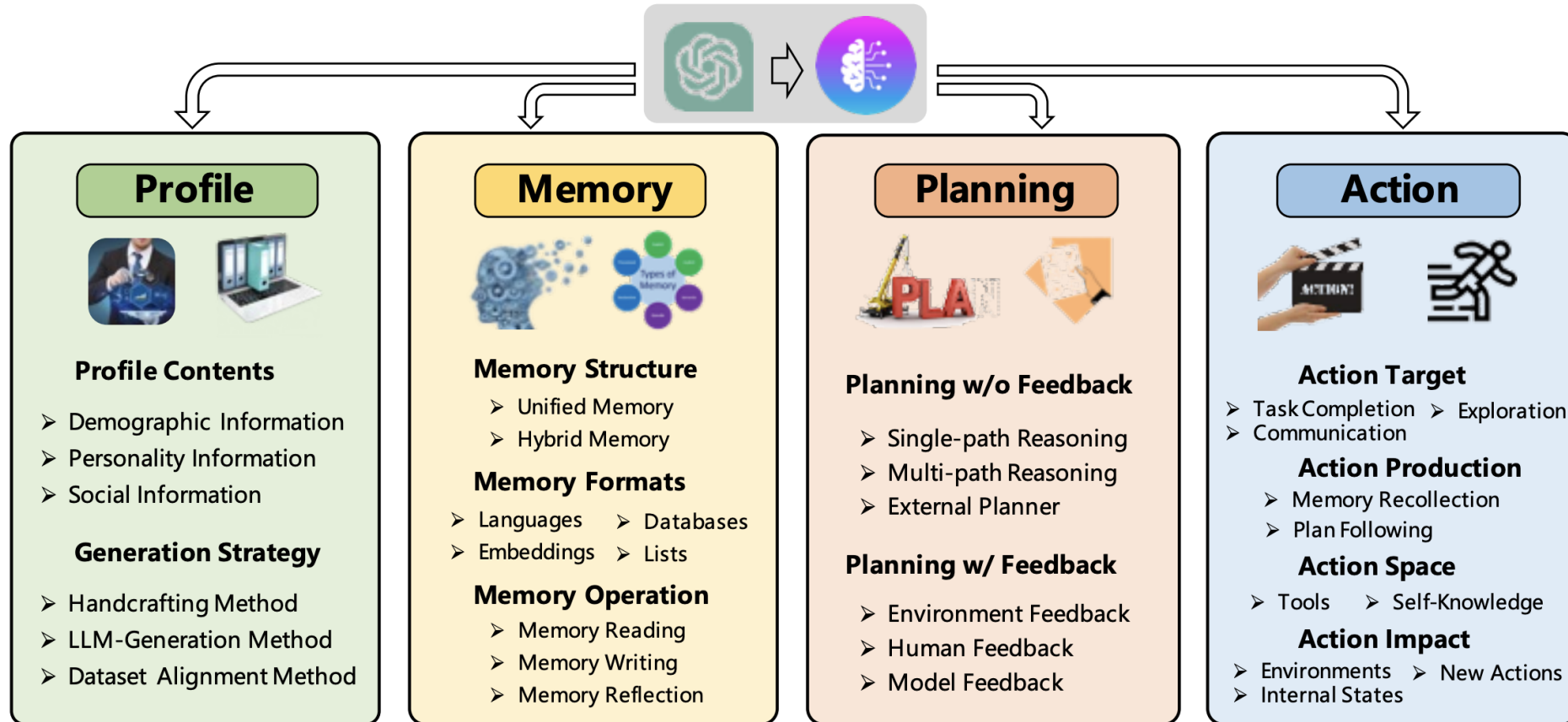
a.k.a. Mechanism Engineering

# LLMs as Agents



<https://www.promptingguide.ai/research/llm-agents>

# LLMs as Agents

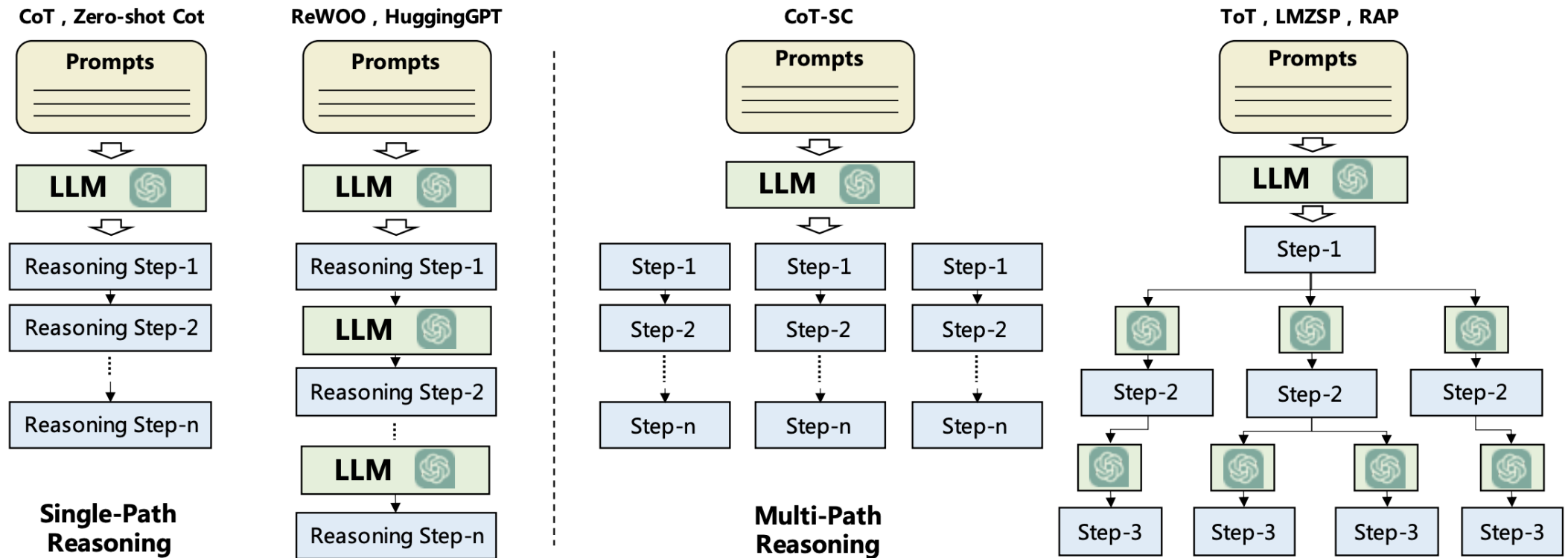


**Fig. 2** A unified framework for the architecture design of LLM-based autonomous agent.



# LLMs as Agents

- Prompting/Planning without feedback



**Fig. 3** Comparison between the strategies of single-path and multi-path reasoning. LMZSP is the model proposed in [54].

# LLMs as Agents

- Prompting/Planning with feedback

(1d) ReAct (Reason + Act)

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

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

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

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

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

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

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

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

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

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

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



Yao et al., “ReAct: Synergizing Reasoning and Acting in Language Models”, 2023.

# LLMs as Agents

- Prompting/Planning with feedback

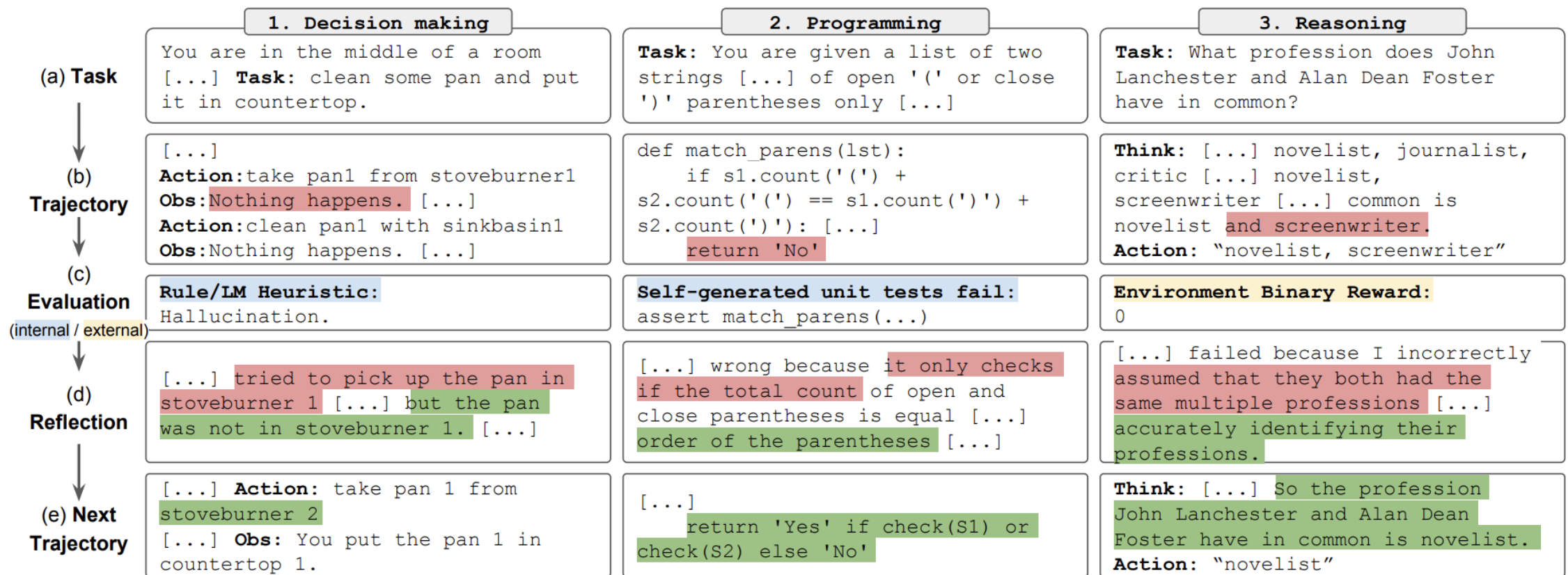
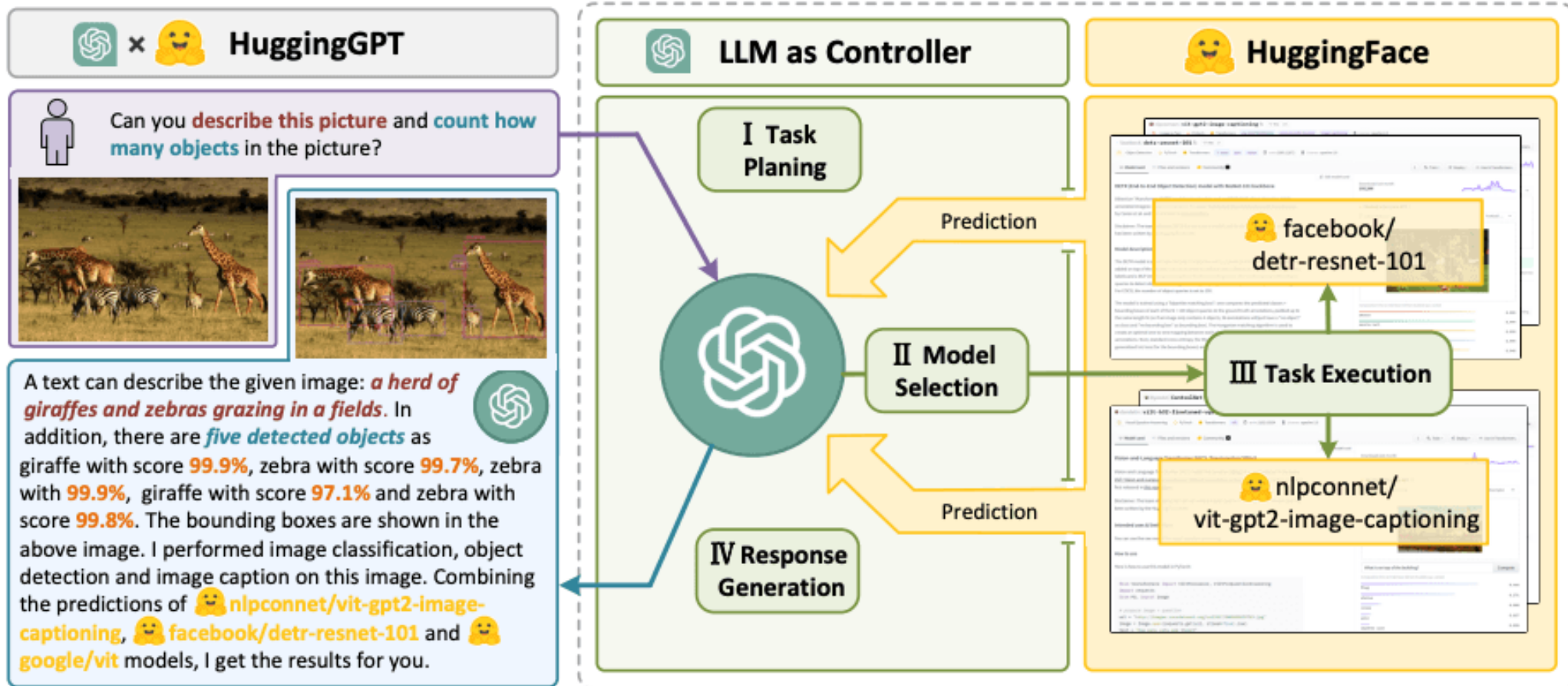


Figure 1: Reflexion works on decision-making 4.1, programming 4.3, and reasoning 4.2 tasks.

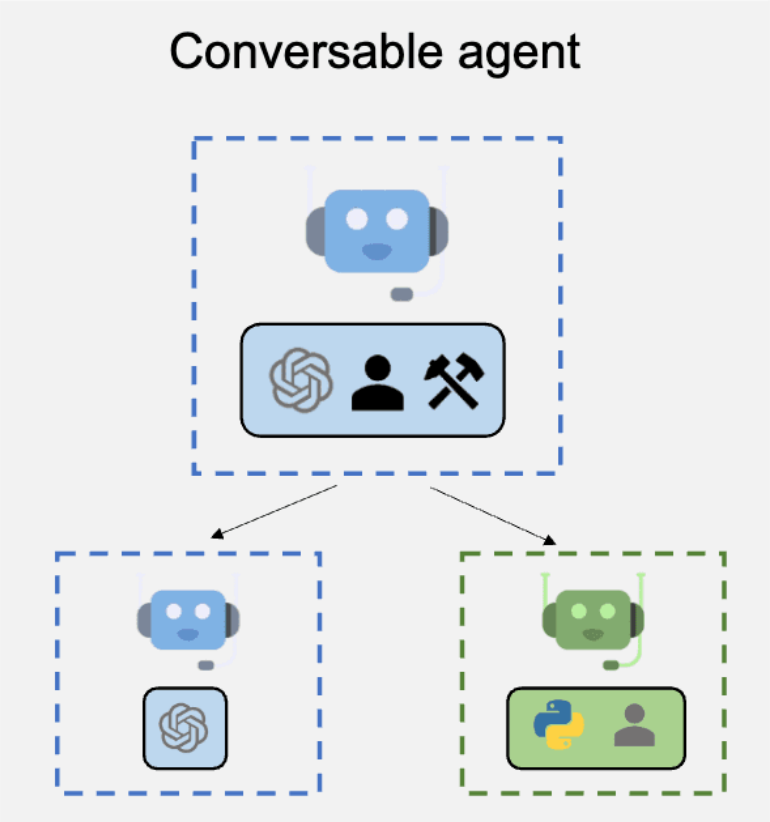
# LLMs as Agents: LLMs and Tools

- MRKL, Toolformer, Function Calling, HuggingGPT, ...



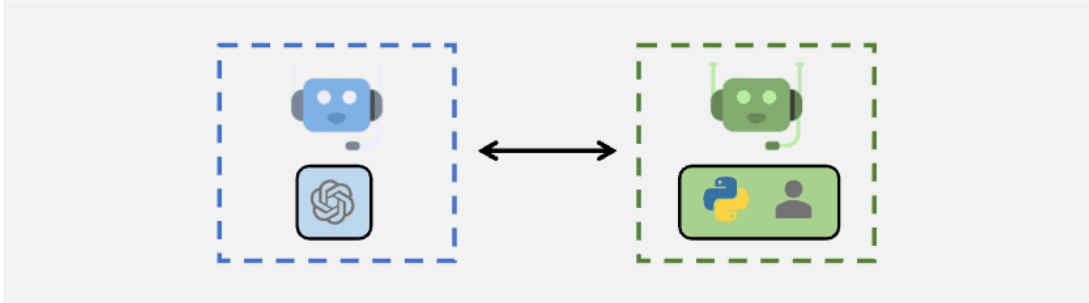
# LLMs as Agents

- Autogen, LangChain, AutoGPT, Langroid, OpenAgents, ....

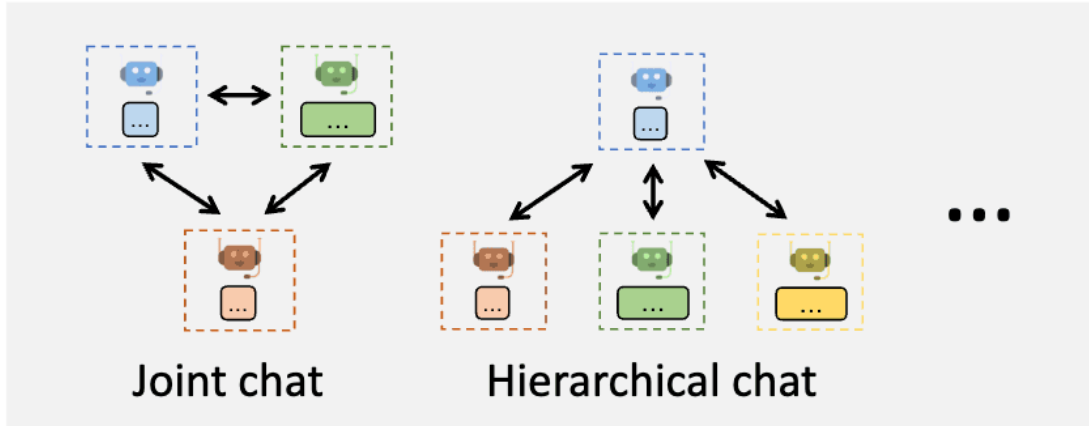


**Agent Customization**

Autogen:



**Multi-Agent Conversations**



**Flexible Conversation Patterns**

## Real-world Challenges

*(On an Ubuntu bash terminal)*  
Recursively set all files in the directory to read-only, except those of mine.

*(Given Freebase APIs)*  
What musical instruments do Minnesota-born Nobel Prize winners play?

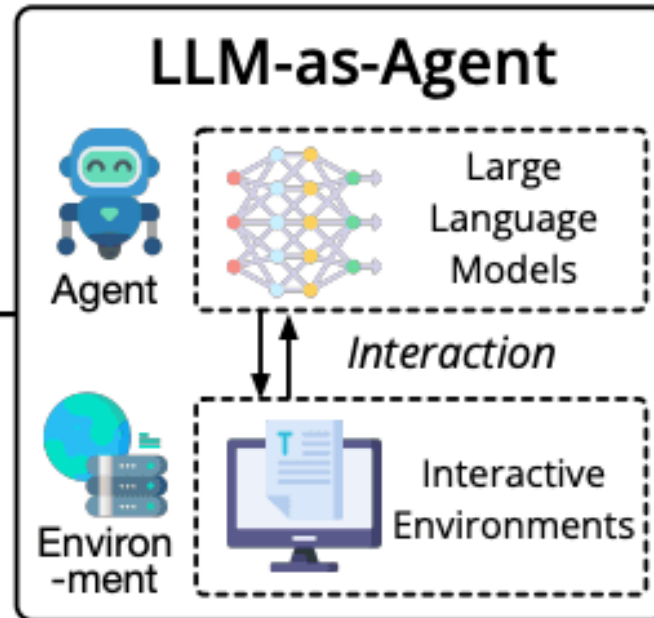
*(Given MySQL APIs and existed tables)*  
Grade students over 60 as PASS in the table.

*(On the GUI of Aquawar)*  
This is a two-player battle game, you are a player with four pet fish cards .....

A man walked into a restaurant, ordered a bowl of turtle soup, and after finishing it, he committed suicide. Why did he do that?

*(In the middle of a kitchen in a simulator)*  
Please put a pan on the dinning table.

*(On the official website of an airline)*  
Book the cheapest flight from Beijing to Los Angeles in the last week of July.



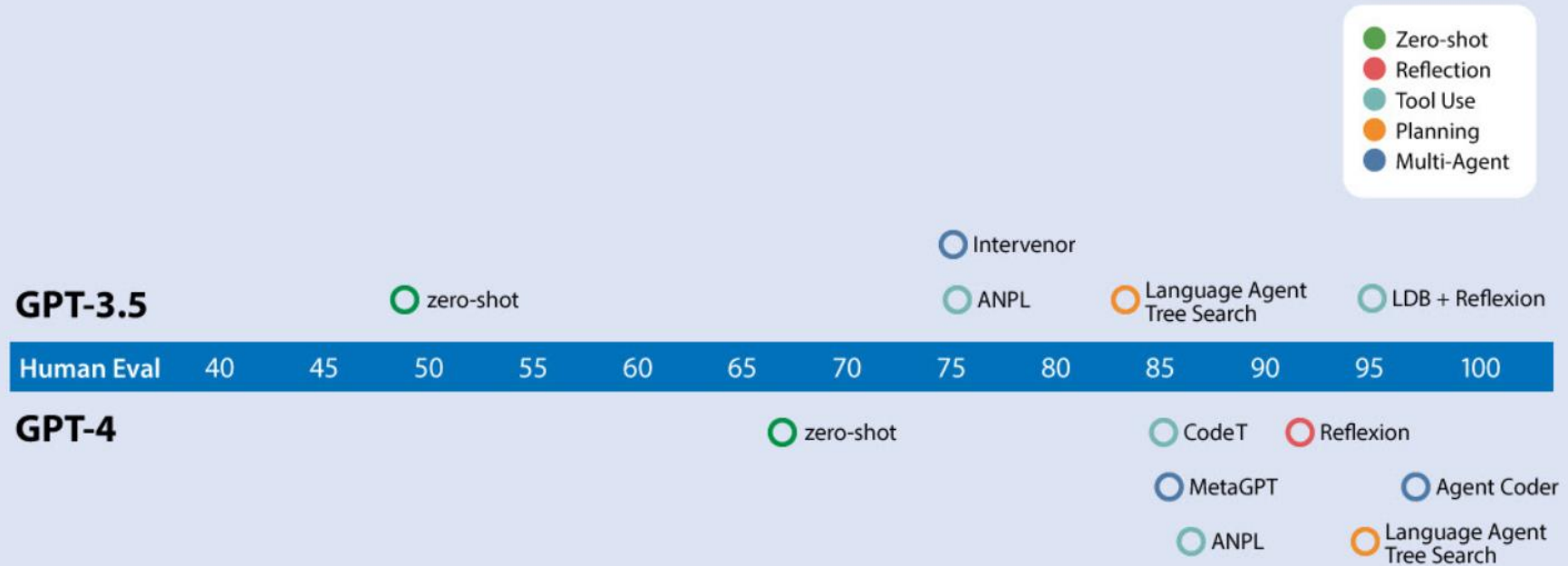
## 8 Distinct Environments



# LLMs as Agents: Example: Anthropic

<https://www.youtube.com/watch?v=vH2f7cjXjKI>

## GPT-3.5 and GPT-4 performance using zero-shot and agent workflows



Performance of GPT-3.5 and GPT-4 (zero-shot) on HumanEval, along with algorithms that use agent workflows on top of GPT-3.5 or GPT-4. Thanks to Joaquin Dominguez and John Santerre for help with this analysis.

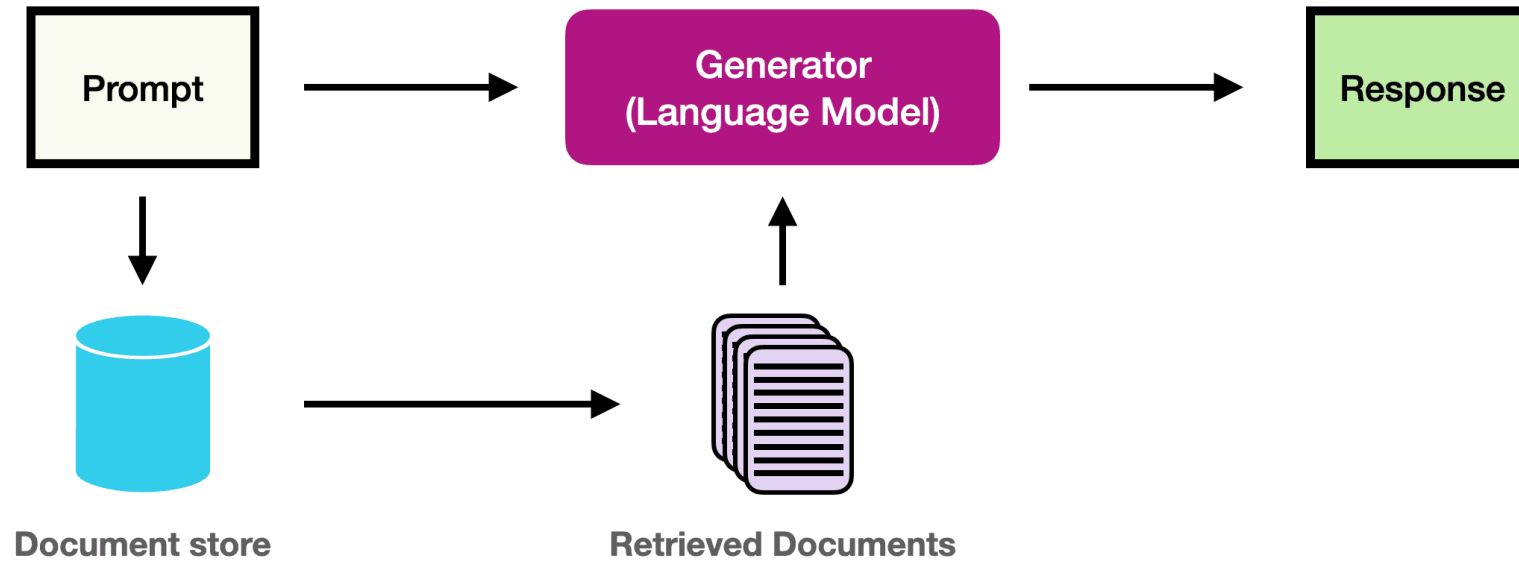
<https://www.deeplearning.ai/the-batch/how-agents-can-improve-llm-performance/>



# LLMs as Agents: Challenges

From <https://www.promptingguide.ai/research/llm-agents>:

- **Role-playing capability:** LLM-based agents typically **need to adapt a role to effectively complete tasks in a domain**. For roles that the LLM doesn't characterize well, it's possible to fine-tune the LLM on data that represent uncommon roles or psychology characters.
- **Long-term planning and finite context length:** planning over a **lengthy history remains a challenge that could lead to errors that the agent may not recover from**. LLMs are also limited in context length they can support which could lead to constraints that limit the capabilities of the agent such as leveraging short-term memory.
- **Generalized human alignment:** it's also challenging to **align agents with diverse human values** which is also common with standard LLMs. A potential solution involves the potential to realign the LLM by designing advanced prompting strategies.
- **Prompt robustness and reliability:** an LLM agent can involve several prompts designed to power the different modules like memory and planning. It's common to **encounter reliability issues in LLMs with even the slightest changes to prompts**. LLM agents involve an entire prompt framework which makes it more prone to robustness issues. The potential solutions include crafting prompt elements through trial and error, automatically optimizing/tuning prompts, or automatically generating prompts using GPT. Another common issue with LLMs is **hallucination which is also prevalent with LLM agents**. These agents rely on natural language to interface with external components that could be introducing conflicting information leading to hallucination and factuality issues.
- **Knowledge boundary:** similar to knowledge mismatch issues that could lead to hallucination or factuality issues, it's challenging to control the knowledge scope of LLMs which can significantly impact the effectiveness of simulations. **Concretely, an LLM's internal knowledge could introduce biases or utilize user-unknown knowledge that could affect the agent's behavior** when operating in specific environments.
- **Efficiency:** LLM agents involve a **significant amount of requests that are handled by the LLM which could affect the efficiency of agent actions because it would depend heavily on the LLM inference speed**. Cost is also a concern when deploying multiple agents.



<https://www.promptingguide.ai/research/llm-agents>

# Retrieval Augmented Generation

# RAG

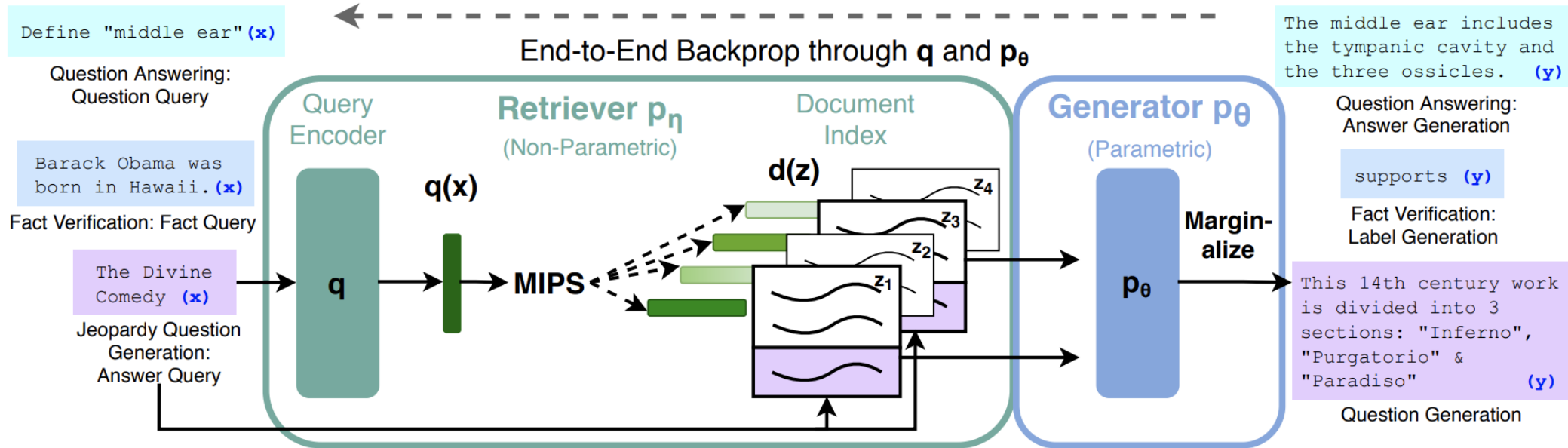


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder + Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query  $x$ , we use Maximum Inner Product Search (MIPS) to find the top-K documents  $z_i$ . For final prediction  $y$ , we treat  $z$  as a latent variable and marginalize over seq2seq predictions given different documents.

# RAG

**RAG-Sequence Model** The RAG-Sequence model uses the same retrieved document to generate the complete *sequence*. Technically, it treats the retrieved document as a single latent variable that is marginalized to get the seq2seq probability  $p(y|x)$  via a top-K approximation. Concretely, the top K documents are retrieved using the retriever, and the generator produces the output sequence probability for each document, which are then marginalized,

$$p_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) p_{\theta}(y|x, z) = \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) \prod_i^N p_{\theta}(y_i|x, z, y_{1:i-1})$$

**RAG-Token Model** In the RAG-Token model we can draw a different latent document for each target *token* and marginalize accordingly. This allows the generator to choose content from several documents when producing an answer. Concretely, the top K documents are retrieved using the retriever, and then the generator produces a distribution for the next output token for each document, before marginalizing, and repeating the process with the following output token, Formally, we define:

$$p_{\text{RAG-Token}}(y|x) \approx \prod_i^N \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) p_{\theta}(y_i|x, z, y_{1:i-1})$$

# RAG

## 2.2 Retriever: DPR

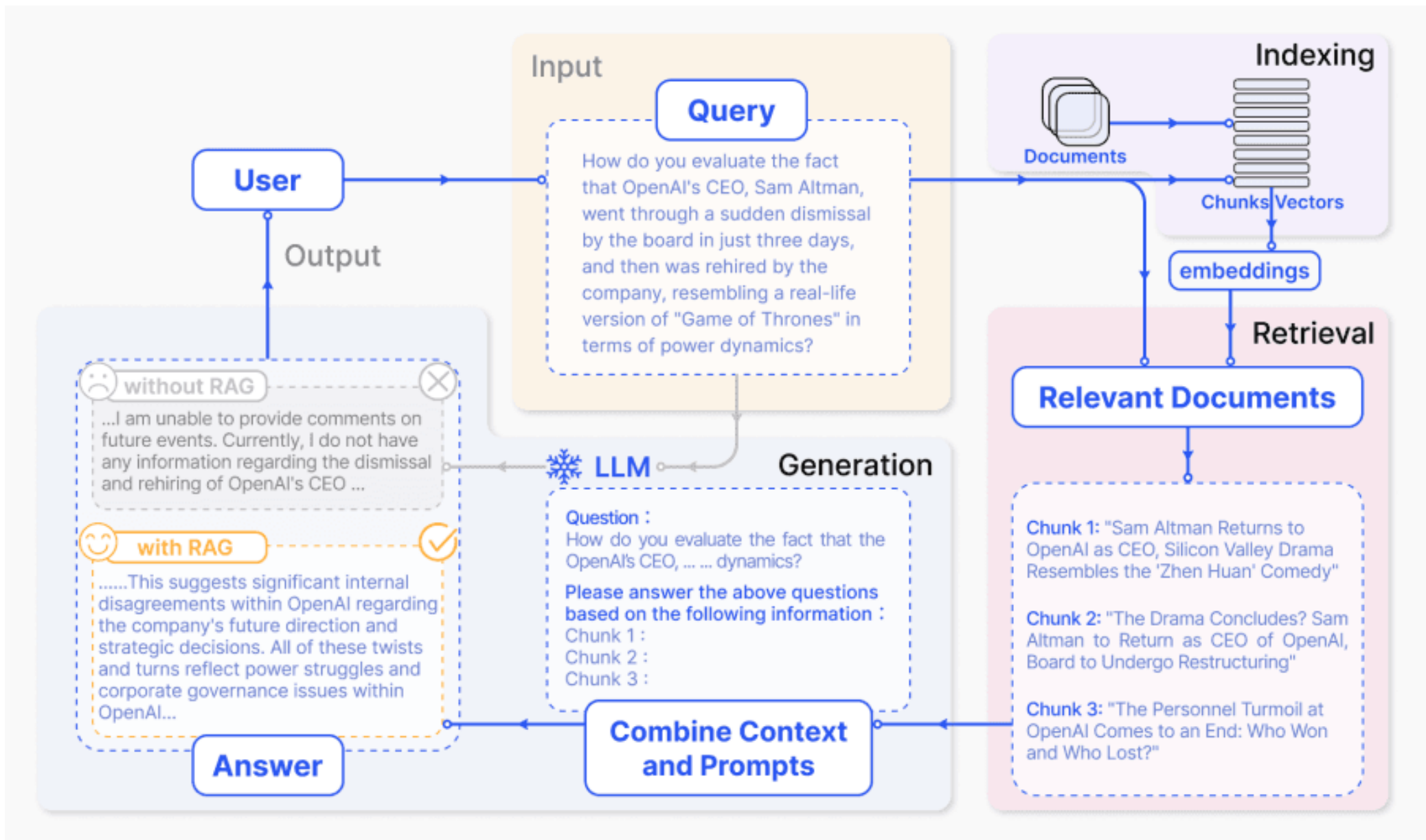
The retrieval component  $p_\eta(z|x)$  is based on DPR [26]. DPR follows a bi-encoder architecture:

$$p_\eta(z|x) \propto \exp(\mathbf{d}(z)^\top \mathbf{q}(x)) \quad \mathbf{d}(z) = \text{BERT}_d(z), \quad \mathbf{q}(x) = \text{BERT}_q(x)$$

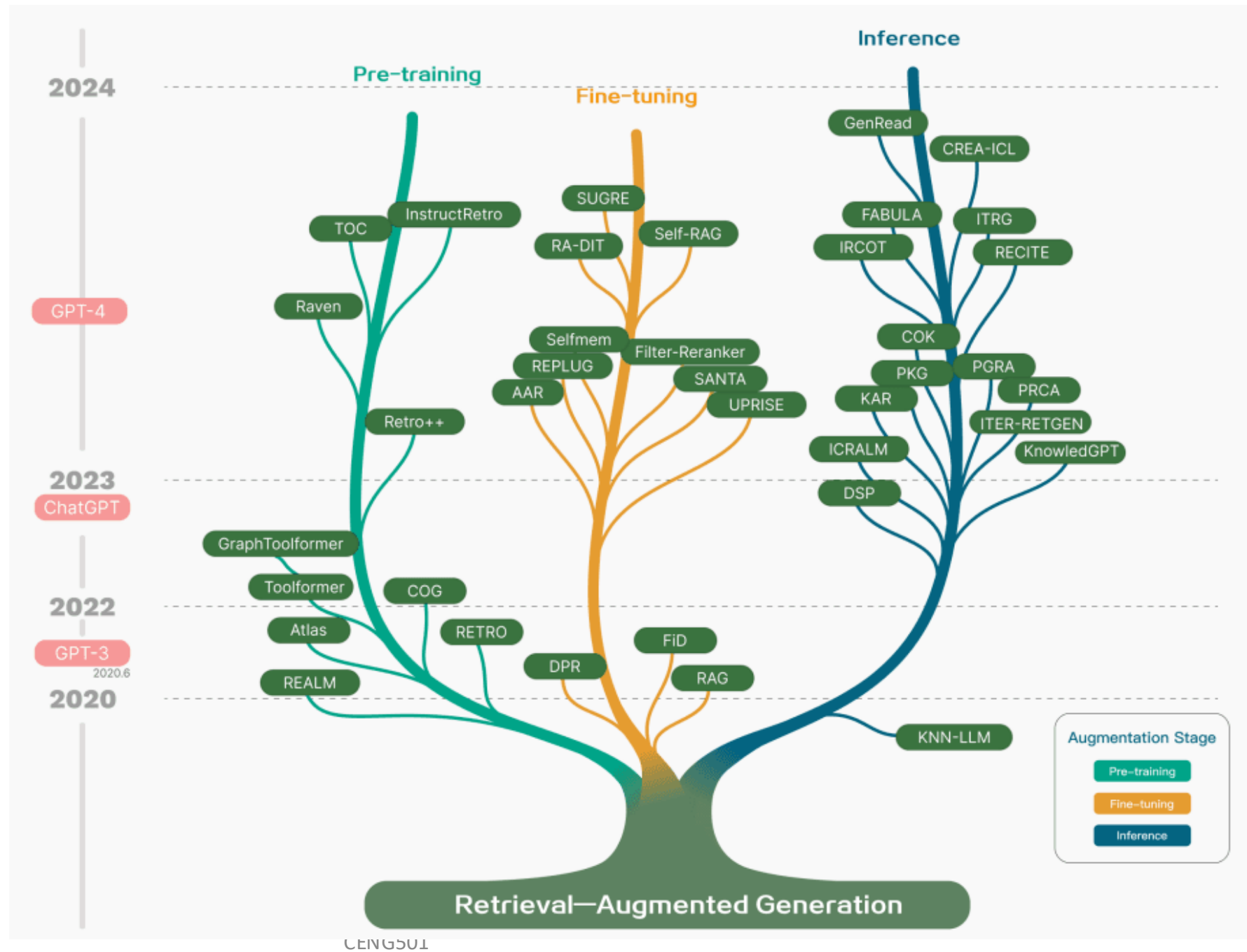
where  $\mathbf{d}(z)$  is a dense representation of a document produced by a  $\text{BERT}_{\text{BASE}}$  *document encoder* [8], and  $\mathbf{q}(x)$  a query representation produced by a *query encoder*, also based on  $\text{BERT}_{\text{BASE}}$ . Calculating  $\text{top-k}(p_\eta(\cdot|x))$ , the list of  $k$  documents  $z$  with highest prior probability  $p_\eta(z|x)$ , is a Maximum Inner Product Search (MIPS) problem, which can be approximately solved in sub-linear time [23]. We use a pre-trained bi-encoder from DPR to initialize our retriever and to build the document index. This retriever was trained to retrieve documents which contain answers to TriviaQA [24] questions and Natural Questions [29]. We refer to the document index as the *non-parametric memory*.

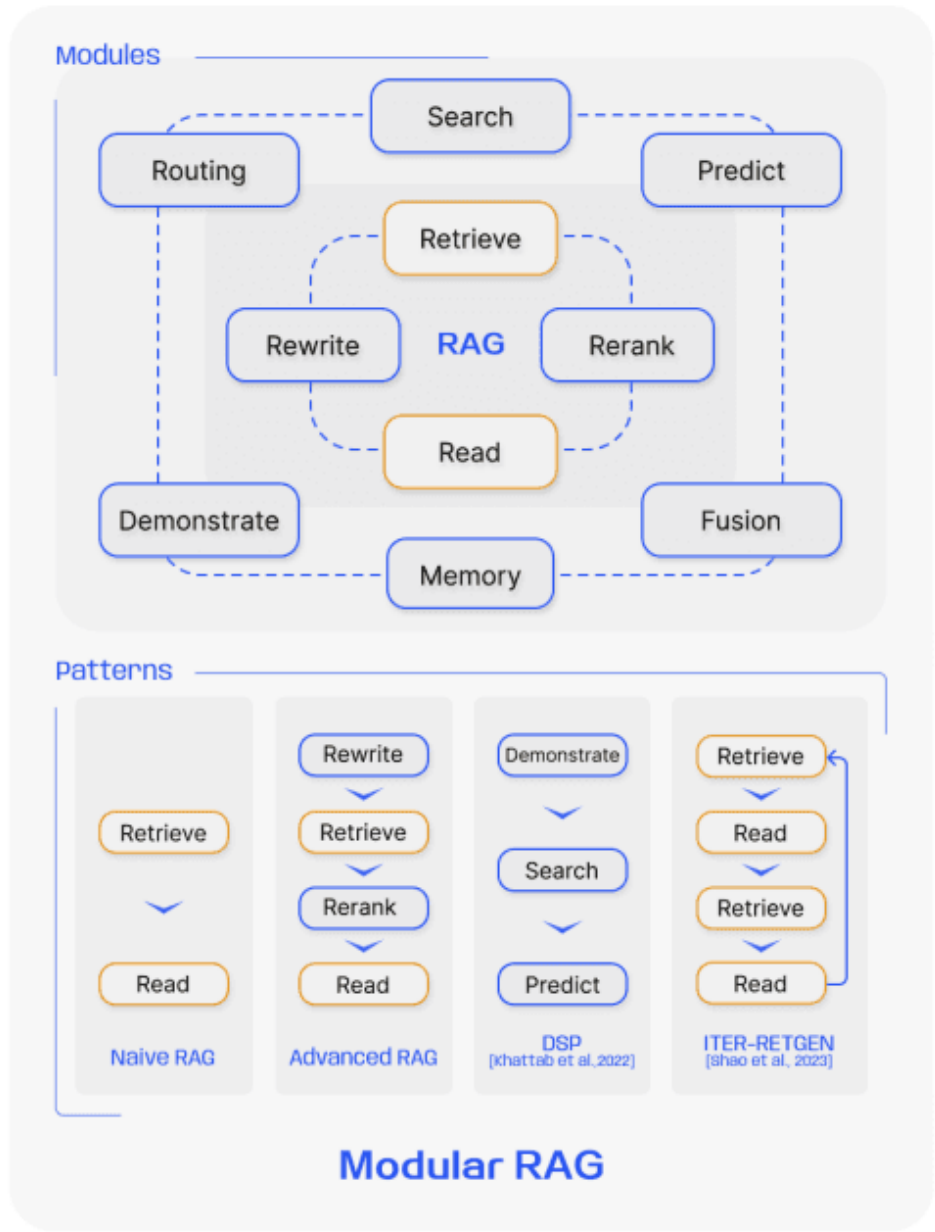
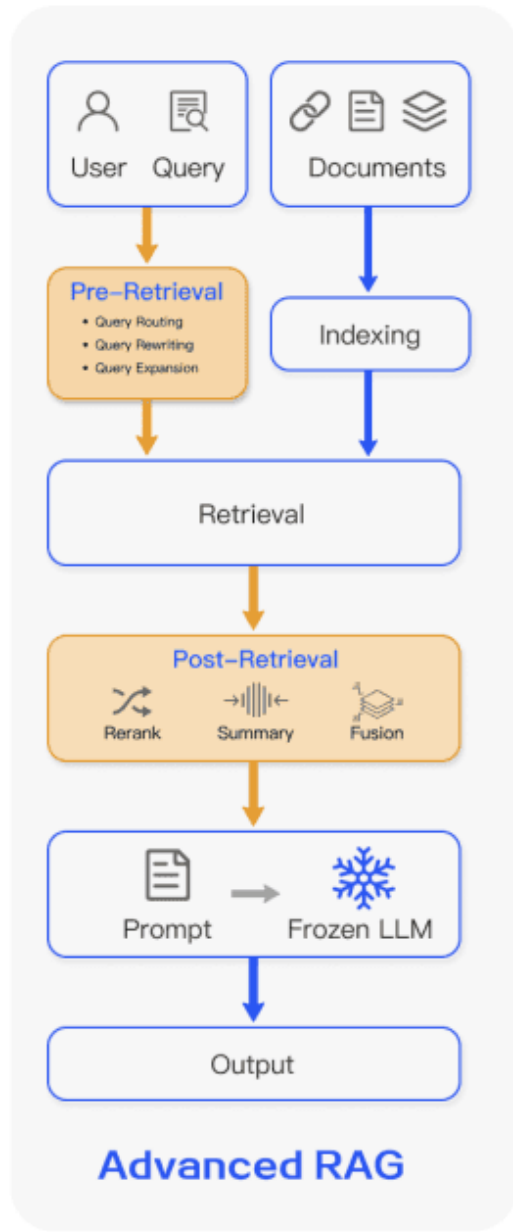
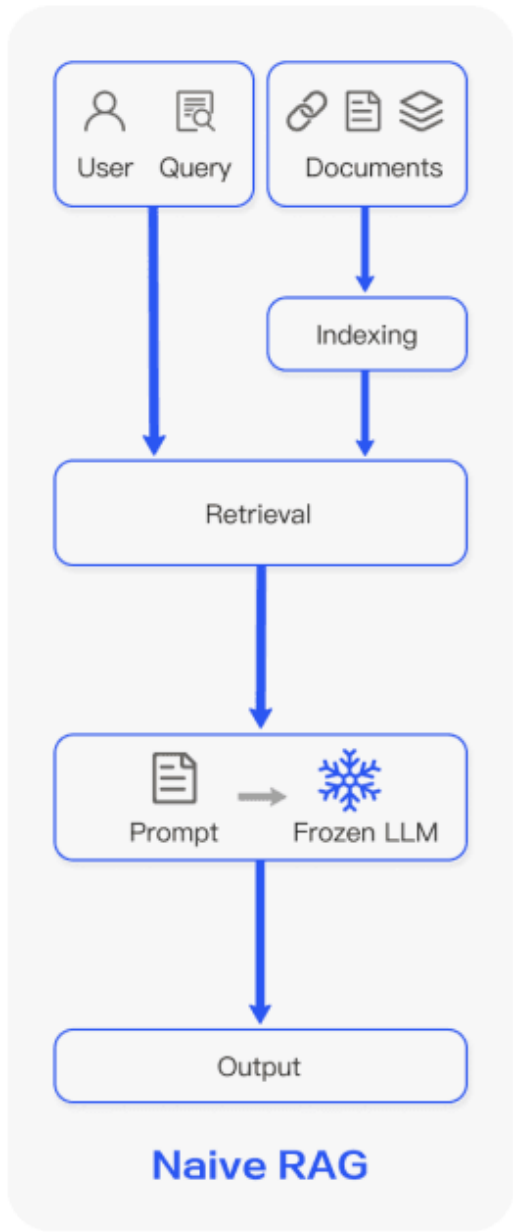
## 2.3 Generator: BART

The generator component  $p_\theta(y_i|x, z, y_{1:i-1})$  could be modelled using any encoder-decoder. We use BART-large [32], a pre-trained seq2seq transformer [58] with 400M parameters. To combine the input  $x$  with the retrieved content  $z$  when generating from BART, we simply concatenate them. BART was pre-trained using a denoising objective and a variety of different noising functions. It has obtained state-of-the-art results on a diverse set of generation tasks and outperforms comparably-sized T5 models [32]. We refer to the BART generator parameters  $\theta$  as the *parametric memory* henceforth.



Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey", 2024.







## » RAG Ecosystem

### Downstream Tasks

Dialogue

Question answering

Summarization

Fact verification

### Technology Stacks

Langchain

LlamaIndex

FlowiseAI

AutoGen

## » The RAG Paradigm

Naive RAG

Advanced RAG

Modular RAG

## » Techniques for Better RAG

Chunk Optimization

Iterative Retrieval

Retriever Fine-tuning

Query Rewriting

Recursive Retrieval

Generator Fine-tuning

Rerank

Adaptive Retrieval

Dual Fine-tuning

## » Key Issues of RAG

What to retrieve

When to retrieve

How to use Retrieval

## » RAG Prospect

### Challenges

Context Length

Robustness

Hybrid

Role of LLMs

Scaling—laws for RAG

Production—ready RAG

### Modality Extension

Image

Audio

Video

Code

### Ecosystem

Customization

Simplification

Specialization

## » Evaluation of RAG

### Evaluation Target

Retrieval Quality

Generation Quality

### Evaluation Aspects

Answer Relevance

Noise Robustness

Context Relevance

Negation Rejection

Answer Faithfulness

Information Integration

Counterfactual Robustness

### Evaluation Framework

Benchmarks

RGB

RECALL

Tools

TruLens

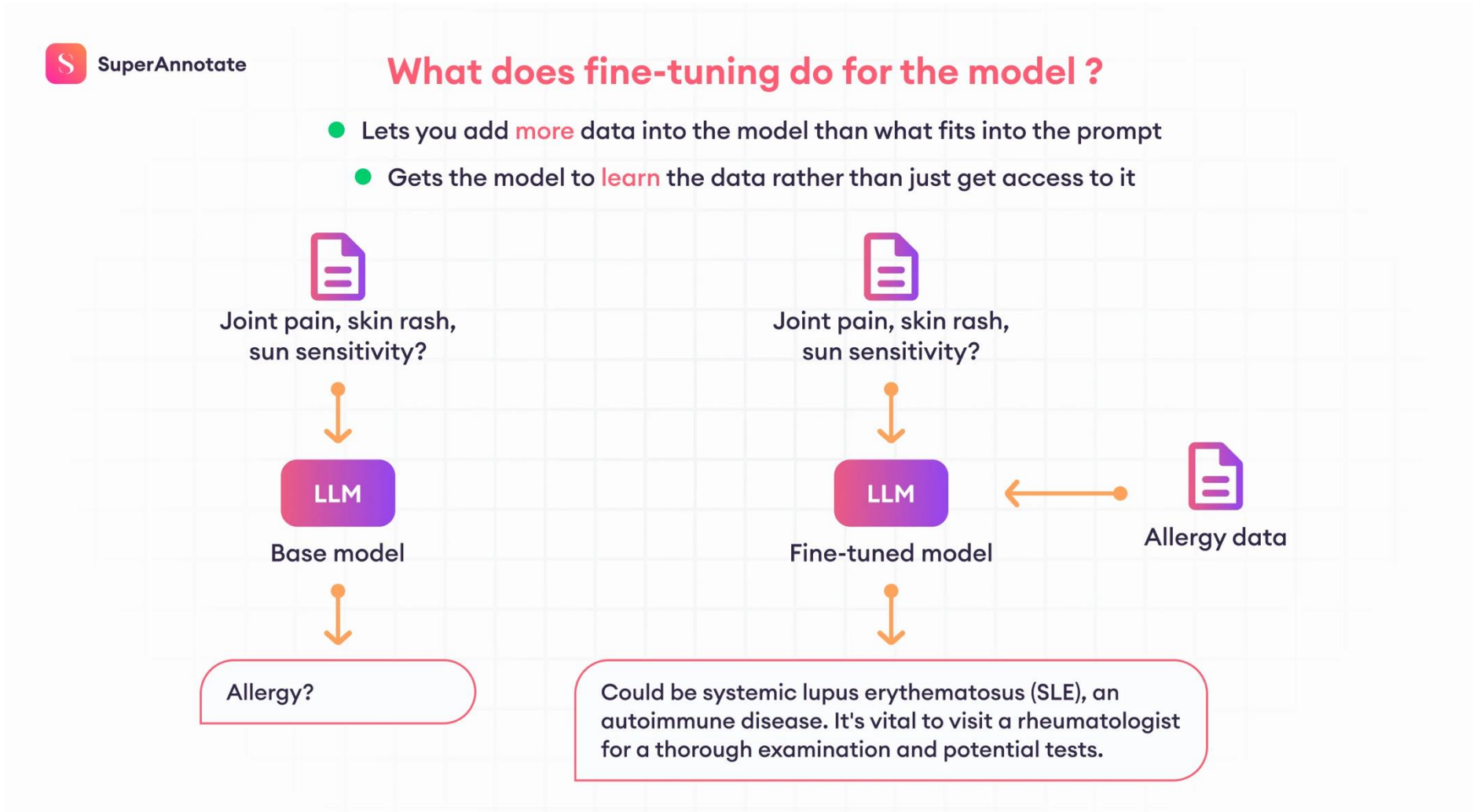
RAGAS

ARES

# Finetuning LLMs

Transfer Learning

# Finetuning an LLM



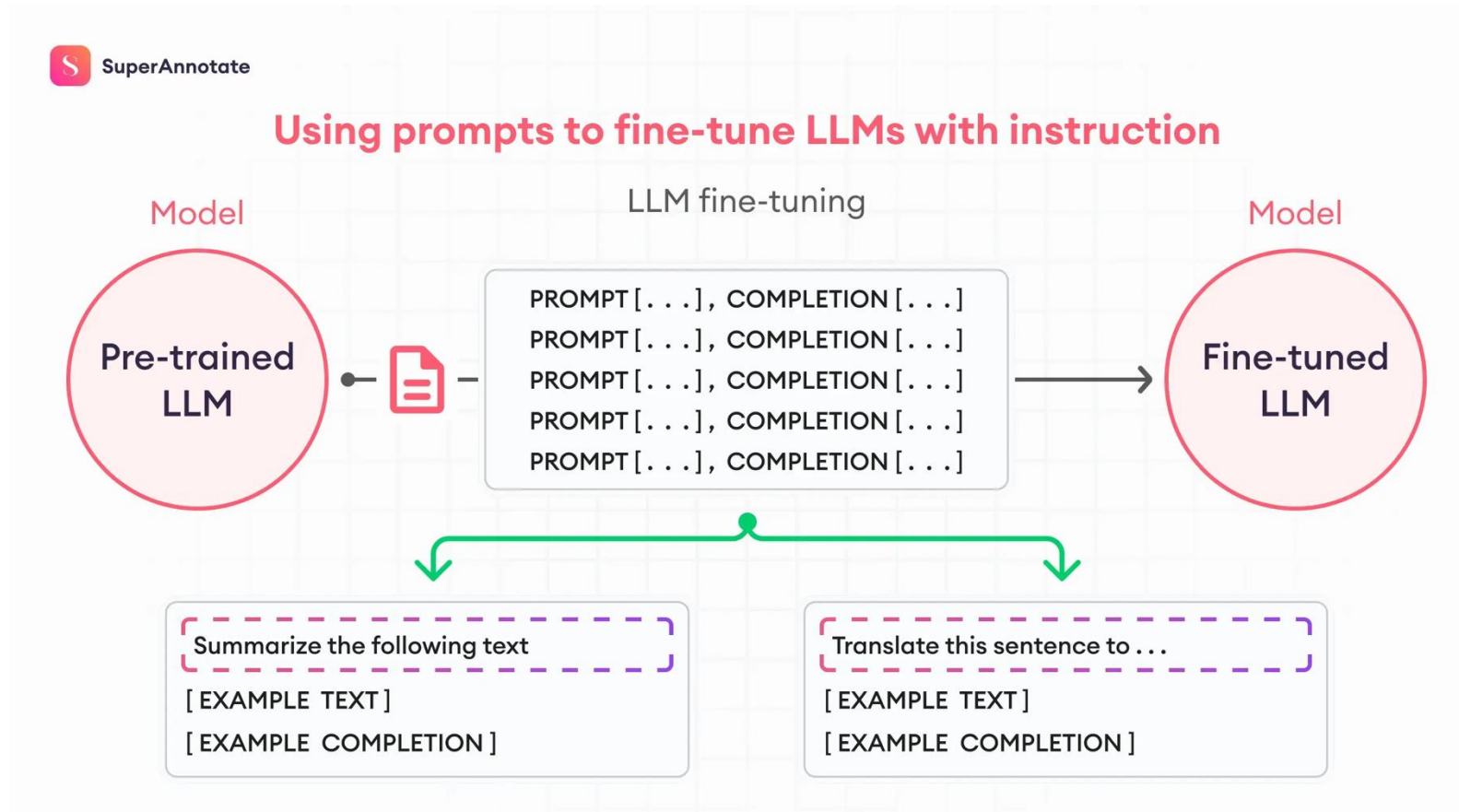
# Finetuning an LLM

Can be helpful in

- specialized applications
- smaller LLMs

# Finetuning an LLM

- Instruction Finetuning

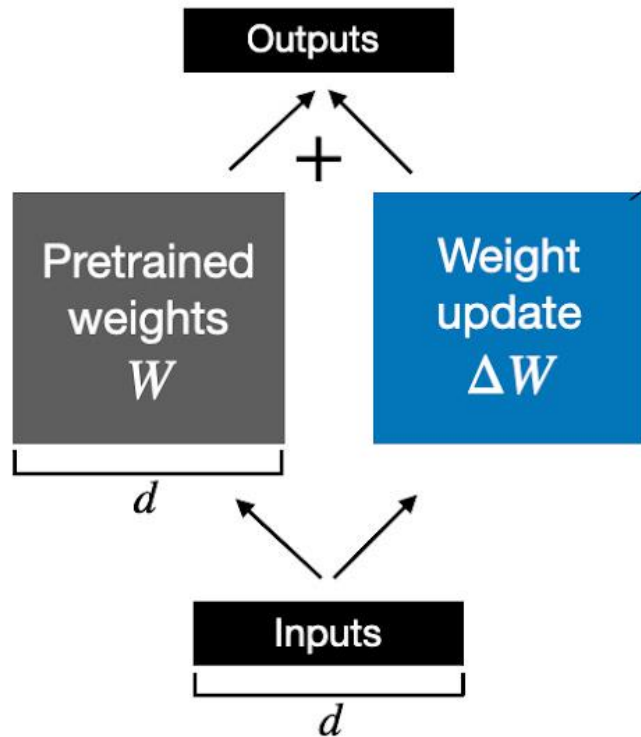


# Finetuning an LLM

- Parameter-efficient Finetuning (vs. Full Finetuning)
  - Update a subset of parameters
- LoRA, LoRA+
- LASER (not finetuning actually)

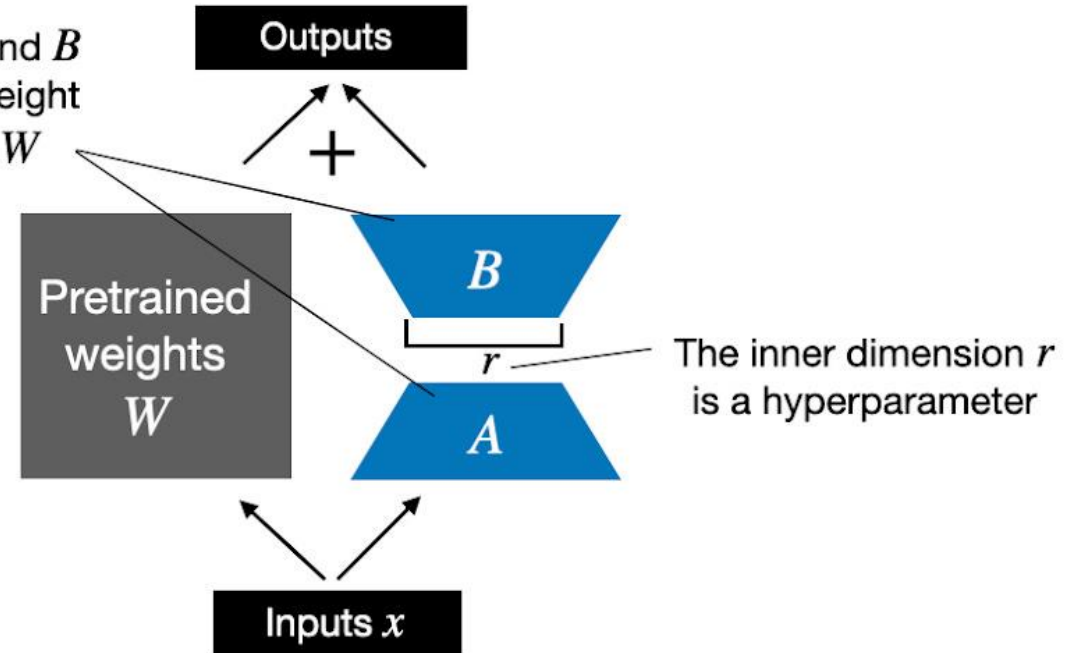
# LoRA

## Weight update in regular finetuning



## Weight update in LoRA

LoRA matrices  $A$  and  $B$  approximate the weight update matrix  $\Delta W$



Target: Attention blocks

# LoRA

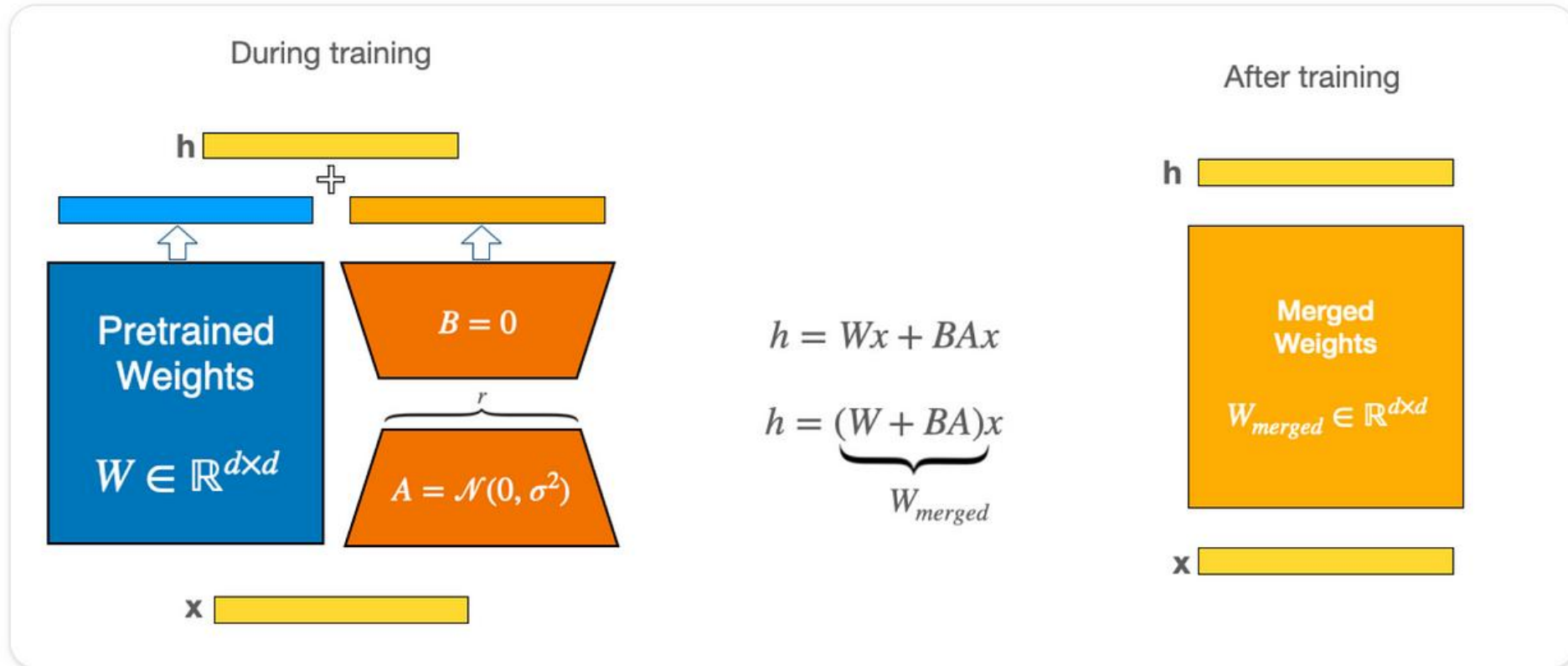


Fig: <https://medium.com/@kailash.thiyagarajan/fine-tuning-large-language-models-with-lora-demystifying-efficient-adaptation-25fa0a389075>



# LoRA

$$\begin{array}{|c|} \hline 1 \\ \hline 3 \\ \hline 7 \\ \hline -4 \\ \hline 2 \\ \hline \end{array} \times \begin{array}{|c|c|c|c|c|} \hline 5 & 1 & -1 & 3 & 4 \\ \hline \end{array} = \begin{array}{|c|c|c|c|c|} \hline 5 & 1 & -1 & 3 & 4 \\ \hline 15 & 3 & -3 & 9 & 12 \\ \hline 35 & 7 & -7 & 21 & 28 \\ \hline -20 & -4 & 4 & -12 & -16 \\ \hline 10 & 2 & -2 & 6 & 8 \\ \hline \end{array}$$

# LoRA

A neural network contains many dense layers which perform matrix multiplication. The weight matrices in these layers typically have full-rank. When adapting to a specific task, Aghajanyan et al. (2020) shows that the pre-trained language models have a low “intrinsic dimension” and can still learn efficiently despite a random projection to a smaller subspace. Inspired by this, we hypothesize the updates to the weights also have a low “intrinsic rank” during adaptation. For a pre-trained weight matrix  $W_0 \in \mathbb{R}^{d \times k}$ , we constrain its update by representing the latter with a low-rank decomposition  $W_0 + \Delta W = W_0 + BA$ , where  $B \in \mathbb{R}^{d \times r}$ ,  $A \in \mathbb{R}^{r \times k}$ , and the rank  $r \ll \min(d, k)$ . During training,  $W_0$  is frozen and does not receive gradient updates, while  $A$  and  $B$  contain trainable parameters. Note both  $W_0$  and  $\Delta W = BA$  are multiplied with the same input, and their respective output vectors are summed coordinate-wise. For  $h = W_0x$ , our modified forward pass yields:

$$h = W_0x + \Delta Wx = W_0x + BAx \quad (3)$$

We illustrate our reparametrization in Figure 1. We use a random Gaussian initialization for  $A$  and zero for  $B$ , so  $\Delta W = BA$  is zero at the beginning of training. We then scale  $\Delta Wx$  by  $\frac{\alpha}{r}$ , where  $\alpha$  is a constant in  $r$ . When optimizing with Adam, tuning  $\alpha$  is roughly the same as tuning the learning rate if we scale the initialization appropriately. As a result, we simply set  $\alpha$  to the first  $r$  we try and do not tune it. This scaling helps to reduce the need to retune hyperparameters when we vary  $r$  (Yang & Hu, 2021).

# LoRA vs Full Fine-tuning: An Illusion of Equivalence

2024

Reece Shuttleworth Jacob Andreas Antonio Torralba Pratyusha Sharma


MIT CSAIL

{rshuttle, jda, torralba, pratyusha}@mit.edu

## ABSTRACT

Fine-tuning is a crucial paradigm for adapting pre-trained large language models to downstream tasks. Recently, methods like Low-Rank Adaptation (LoRA) have been shown to match the performance of fully fine-tuned models on various tasks with an extreme reduction in the number of trainable parameters. Even in settings where both methods learn similarly accurate models, *are their learned solutions really equivalent?* We study how different fine-tuning methods change pre-trained models by analyzing the model’s weight matrices through the lens of their spectral properties. We find that full fine-tuning and LoRA yield weight matrices whose singular value decompositions exhibit very different structure; moreover, the fine-tuned models themselves show distinct generalization behaviors when tested outside the adaptation task’s distribution. More specifically, we first show that the weight matrices trained with LoRA have new, high-ranking singular vectors, which we call *intruder dimensions*. Intruder dimensions do not appear during full fine-tuning. Second, we show that LoRA models with intruder dimensions, despite achieving similar performance to full fine-tuning on the target task, become worse models of the pre-training distribution and adapt less robustly to multiple tasks sequentially. Higher-rank, rank-stabilized LoRA models closely mirror full fine-tuning, even when performing on par with lower-rank LoRA models on the same tasks. These results suggest that models updated with LoRA and full fine-tuning access different parts of parameter space, even when they perform equally on the fine-tuned distribution. We conclude by examining why intruder dimensions appear in LoRA fine-tuned models, why they are undesirable, and how their effects can be minimized.

## LoRA Learns Intruders

 *Intruder Dimensions* are singular vectors that are dissimilar to those in the pre-trained weight matrix and are learned during fine-tuning.

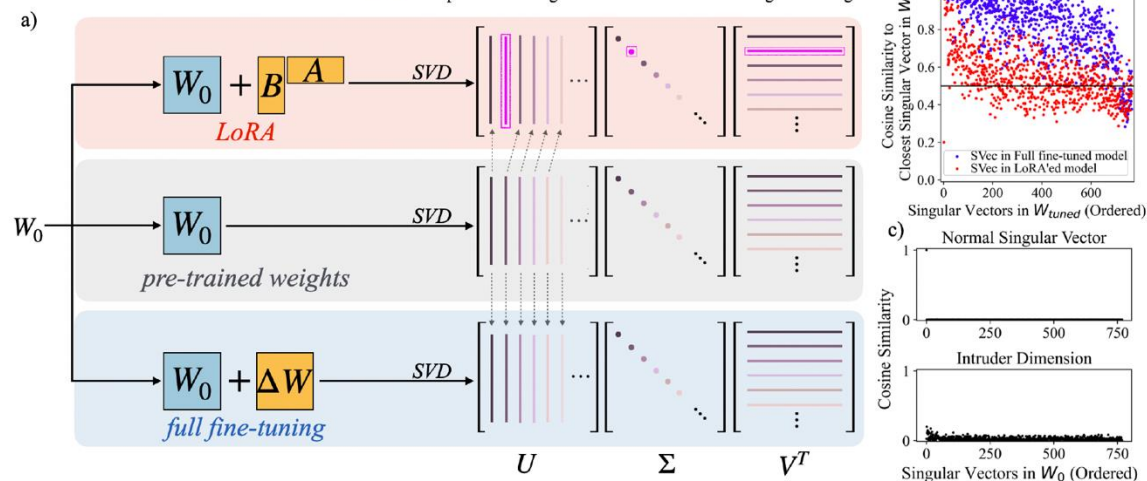


Figure 2: **Characterizing structural differences between solutions learnt by LoRA Vs full Fine-tuning.** **a)** We measure the changes to the SVD of the pre-trained weights made during fine-tuning. We observe *intruder dimensions* introduced by LoRA in top ranking singular vectors but by full fine-tuning. **b)** Comparing a matrix fine-tuned with full fine-tuning or LoRA. **c)** Comparing a normal singular vs an intruder dimension to all pre-trained singular vectors.

# LoRA+

## Abstract

In this paper, we show that Low Rank Adaptation (LoRA) as originally introduced in (Hu et al., 2021) leads to suboptimal finetuning of models with large width (embedding dimension). This is due to the fact that adapter matrices  $A$  and  $B$  in LoRA are updated with the same learning rate. Using scaling arguments for large width networks, we demonstrate that using the same learning rate for  $A$  and  $B$  does not allow efficient feature learning. We then show that this suboptimality of LoRA can be corrected simply by setting different learning rates for the LoRA adapter matrices  $A$  and  $B$  with a well-chosen fixed ratio. We call this proposed algorithm LoRA+. In our extensive experiments, LoRA+ improves performance (1% – 2% improvements) and finetuning speed (up to  $\sim 2X$  SpeedUp), at the same computational cost as LoRA.

Hayou et al., “LoRA+: Efficient Low Rank Adaptation of Large Models”, 2024.

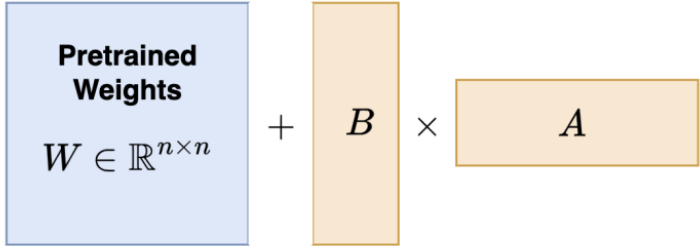
	LoRA	LoRA+
Parameterization		
Training	$A \leftarrow A - \eta \times G_A$ $B \leftarrow B - \eta \times G_B$	$A \leftarrow A - \eta \times G_A$ $B \leftarrow B - \lambda \eta \times G_B$ $\lambda \gg 1$

Figure 1. The key difference between standard LoRA and LoRA+ is in how learning rates are set (the matrices  $G_A$  and  $G_B$  are ‘effective’ gradients from AdamW) With standard LoRA, the learning rate is the same for  $A$  and  $B$ , which provably leads to suboptimal learning when embedding dimension is large. In LoRA+, we set the learning rate of  $B$  to be  $\lambda \times$  that of  $A$ , where  $\lambda \gg 1$  is fixed. We later provide guidelines on how to set  $\lambda$ .





# LASER

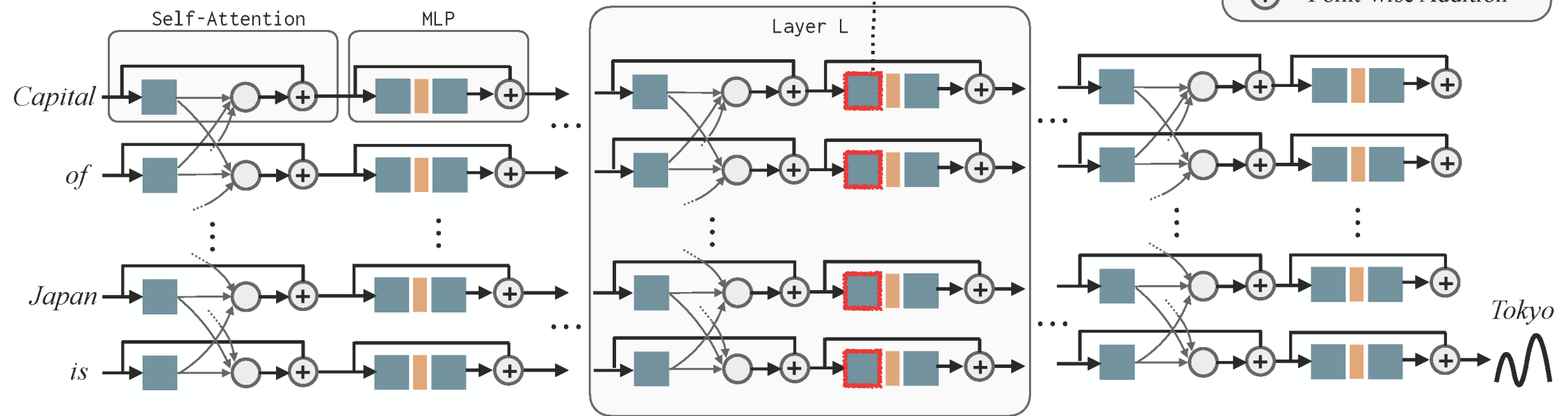
Sharma et al., "The Truth is in There: Improving Reasoning in Language Models with Layer-Selective Rank Reduction", 2023.

## LASER

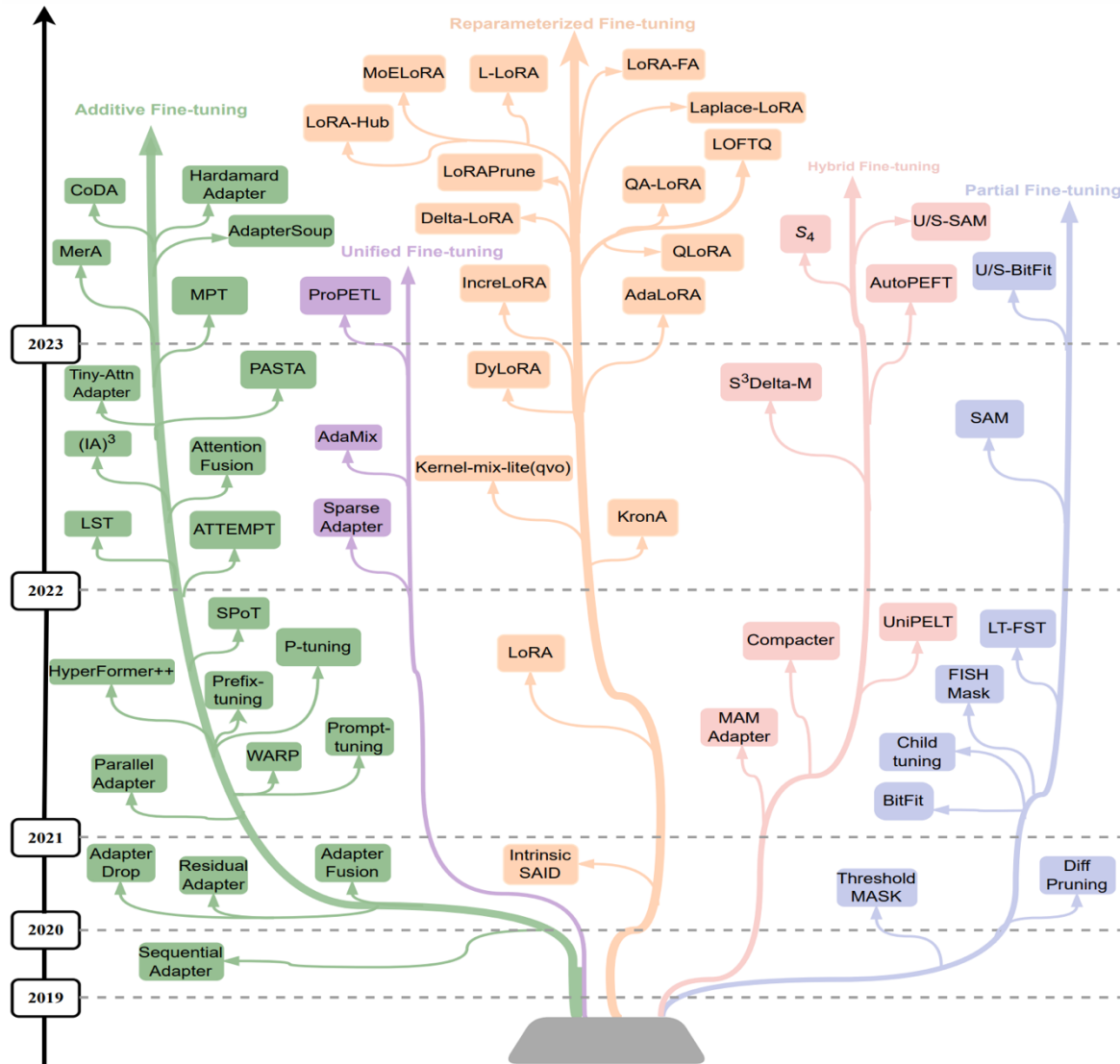
Replace  $W$  in specific layers with its low rank approximation  $W_{LR}$

$$W \rightarrow W_{LR} = U \Sigma V^T$$

-  Linear Layer
-  Self-Attention Layer
-  Nonlinear activation
-  Point-wise Addition



# Parameter Efficient Fine Tuning (PEFT)



Xu et al., "Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment", 2023.

# Finetuning an LLM: Challenges

1. **Overfitting:** Fine-tuning can be prone to overfitting, a condition where the model becomes overly specialized on the training data and performs poorly on unseen data. This risk is particularly pronounced when the task-specific dataset is small or not representative of the broader context.
2. **Catastrophic Forgetting:** During fine-tuning for a specific task, the model may forget previously acquired general knowledge. This phenomenon, known as catastrophic forgetting, can impair the model's adaptability to diverse tasks.
3. **Bias Amplification:** Pre-trained models inherit biases from their training data, which fine-tuning can inadvertently amplify when applied to task-specific data. This amplification may lead to biased predictions and outputs, potentially causing ethical concerns.
4. **Generalization Challenges:** Ensuring that a fine-tuned model generalizes effectively across various inputs and scenarios is challenging. A model that excels in fine-tuning datasets may struggle when presented with out-of-distribution data.
5. **Data Requirements:** Fine-tuning necessitates task-specific labelled data, which may not always be available or clean. Inadequate or noisy data can negatively impact the model's performance and reliability.
6. **Hyperparameter Tuning Complexity:** Selecting appropriate hyperparameters for fine-tuning can be intricate and time-consuming. Poor choices may result in slow convergence, overfitting, or suboptimal performance.
7. **Domain Shift Sensitivity:** Fine-tuning data significantly different from the pre-training data can lead to domain shift issues. Addressing this problem often requires domain adaptation techniques to bridge the gap effectively.
8. **Ethical Considerations:** Fine-tuned large language models may inadvertently generate harmful or inappropriate content, even when designed for benign tasks. Ensuring ethical behaviour and safety is an ongoing challenge, necessitating responsible AI practices.
9. **Resource Intensiveness:** Fine-tuning large models demands substantial computational resources and time, posing challenges for smaller teams or organizations with limited infrastructure and expertise.
10. **Unintended Outputs:** Fine-tuning cannot guarantee that the model consistently produces correct or sensible outputs. It may generate plausible but factually incorrect responses, requiring vigilant post-processing and validation.
11. **Model Drift:** Over time, a fine-tuned model's performance can deteriorate due to changes in data distribution or the evolving environment. Regular monitoring and re-fine-tuning may become necessary to maintain optimal performance and adapt to evolving conditions.

# Finetuning vs. RAG

## Factors to consider

- Nature of the task: Specialized tasks might benefit from finetuning. RAG is better for problems requiring external / up-to-date knowledge.
- Data availability: Finetuning requires a lot of data. RAG can utilize existing data.
- Resource: Finetuning can be expensive. RAG is easy to integrate.



# Learning from Model