

# Large Language Models

AI602: Recent Advances in Deep Learning  
Lecture 2

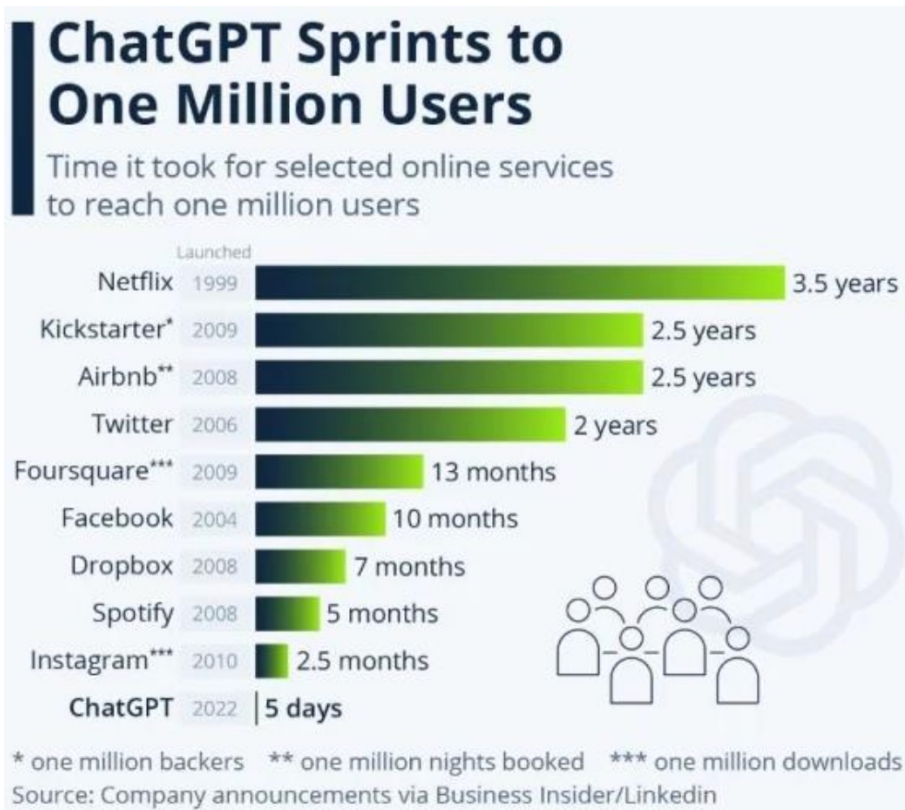
Slide made by

Jaehyung Kim, Hyunseok Lee, Dongyoung Kim  
KAIST AI

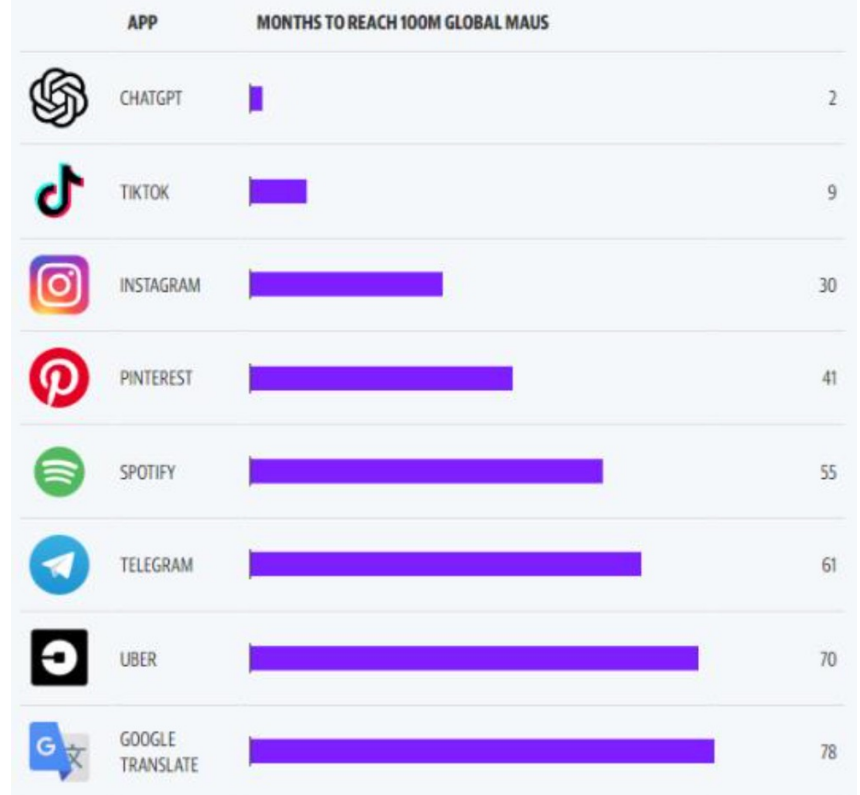
# Introduction

## Impact of ChatGPT

- ChatGPT sets record for **fastest-growing** user-base service
  - 5 days for 1M users and 2 months for 100M users, respectively



## HOW LONG IT TOOK TOP APPS TO HIT 100M MONTHLY USERS



## Impact of ChatGPT

- ChatGPT sets record for **fastest-growing** user-base service
- ChatGPT can generate **realistic texts for complex domains**
  - E.g., New York City School bans ChatGPT amid cheating worries
  - E.g., Discussions to use ChatGPT to write academic papers and lists on the authors

### 뉴욕시 교육국, 챗봇 사용금지 조치

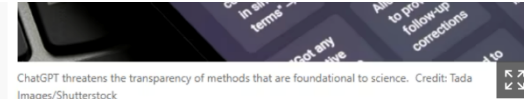
교육국 장비와 공립교 인터넷 네트워크서  
인공지능 챗봇 '챗GPT' 프로그램 접근 차단  
"부정행위 우려, 비판적 사고 능력 발달 저해"

뉴욕시 교육국이 교육국 교육장비(랩톱, 아이패드 등)와 공립교 인터넷 네트워크에서 인공지능(이하 AI) 챗봇 '챗GPT(ChatGPT)' 사용을 금지한다고 밝혔다.

3일 교육국은 해당 프로그램이 "학생들의 학습에 부정적인 영향을 미치고, 콘텐츠의 안전과 정확성에 대한 우려"를 이유로 프로그램에 대한 접근을 차단한다고 밝혔다. 특히, "해당 프로그램이 학생들의 비판적 사고 및 문제해결 능력을 기르는데 방해된다"고 지적했다.

챗GPT는 지난해 11월 인공지능 연구 기업인 오픈AI에서 공개한 AI 챗봇 서비스로 단순한 대화 답변을 넘어, 실질적인 가치를 담은 콘텐츠를 스스로 생산할 수 있다는 가능성을 보여주고 있어 주목받고 있다.

이런 기술 자체가 새롭지는 않지만, 챗GPT는 '더 인간 같은' 수준 높은 글을 작성할 수 있어 학생들이 집에서 숙제나 온라인 시험을 치를 때 활용해도 교사가 모를 가능성이 커 부정행위 등 사회적인 문제로 부상할 수도 있다는 분석이 나온다.



ChatGPT threatens the transparency of methods that are foundational to science. Credit: Tada Images/Shutterstock

네이처와 네이처의 출판사 스프링거 네이처는 24일(현지 시간) "챗GPT를 포함한 AI를 논문 저자로 인정하지 않을 것"이라며 사실을 통해 가이드를 발표했다./ 네이처 뉴스 사실 캡처

#### ◇ 학술계에서도 '챗GPT는 도구' vs '무조건 제한' 엇갈려

실제로 연구 현장에선 일부 연구자들 중심으로 챗GPT의 연구 역량을 미리 예상한 듯 챗GPT를 연구에 사용하고 공동 저자로 지정하고 있다. 지난달 12일 의학논문 사전 공개사이트인 메드아카이브(MedRxiv)에는 챗GPT를 세 번째 공저자로 한 논문이 발표됐다.

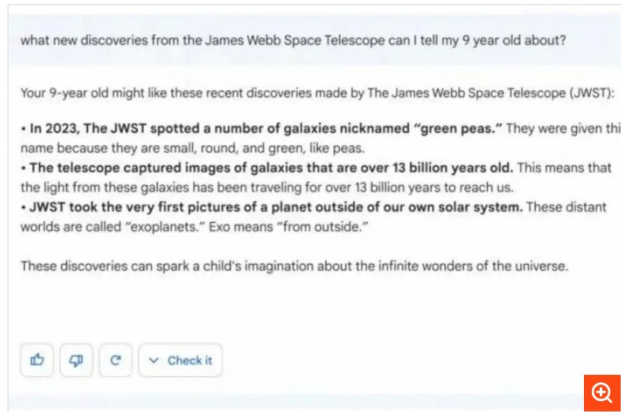
학계와 학술 출판계는 챗GPT를 학술 논문 저자로 인정할 것인가를 두고 논란이 여전히 계속되고 있다.

국제학술지 네이처를 발간하는 스프링거 네이처는 24일 "챗GPT를 포함한 AI를 논문 저자로 인정하지 않겠다"며 "AI가 쓴 글을 잡아내기 위한 기술을 개발하고 있다"고 밝혔다. 네이처는 다만 "챗GPT같은 AI를 연구에 활용하는 경우에는 논문에 명시해야 한다"는 가이드 라인을 내놨다. 저자는 아니지만 연구 도구로서 챗GPT 사용은 인정할 셈이다. 전문가들은 스프링거 네이처가 과학, 기술, 의학 등 3000종 이상의 학술지를 출판하는 대형 학술 출판기업인만큼 이 같은 조치가 학계에 미칠 영향이 클 것으로 보고 있다.

## Impact of ChatGPT

- ChatGPT sets record for **fastest-growing** user-base service
- ChatGPT can generate **realistic texts for complex domains**
- ChatGPT can serve as a **new effective search engine**
  - Microsoft announces that ChatGPT will be incorporated on Bing
  - Google release Bard, google's generative search engine, similar to ChatGPT

일반 사용자용 AI 플랫폼 출시를 위해 '코드 레드'를 선언한 것으로 알려진 구글도 곧 대열에 합류한다. 6일(현지시간) 구글 CEO 순다르 피차이가 공개한 **바드(Bard)**는 ChatGPT처럼 크고 작은 질문에 대해 자세한 답변을 생성하는 대화형 AI다.



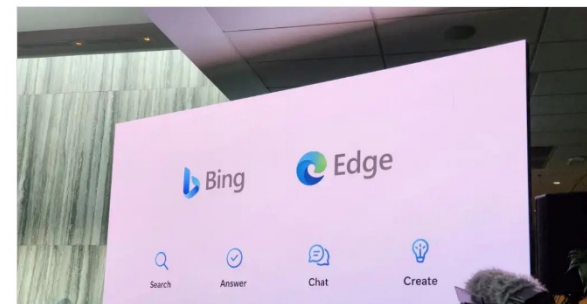
구글은 미묘한 질문에 대한 바드의 답변을 공개했다. 9세 아동 수준에 맞는 방식으로 제임스 웹(James Webb) 우주 망원경을 설명하는 방법에 대한 답이다. © Google

## 마이크로소프트, ChatGPT 통합한 새로워진 '빙' 공개...엣지에도 AI 적용

Mark Hachman | PCWorld 7일 전

7일(현지시간) 마이크로소프트가 미국 워싱턴주 레드먼드에 있는 본사에서 언론 행사를 열고 ChatGPT 기능이 도입된 새로운 빙을 공개했다. 마이크로소프트는 새로운 빙을 "웹을 위한 부조종사"라고 표현했다.

마이크로소프트는 빙에 자체적인 버전의 ChatGPT 알고리즘으로 구동되는 컨텍스트 검색 기능을 적용한다. 또한 각주 링크를 제공하는 채팅 인터페이스도 빙 검색 결과에 추가된다. 엣지 브라우저에도 AI를 통합해 재무 수익 보고서를 요약하는 등의 작업을 요청할 수 있다. 빙의 새로운 검색 엔진 인터페이스는 현재 일부 사용자를 대상으로만 배포됐으며, 수주 내 전체 사용자에게 확장될 예정이다.



So, what is ChatGPT?

## What is ChatGPT?

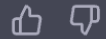
- Ask to ChatGPT itself

J

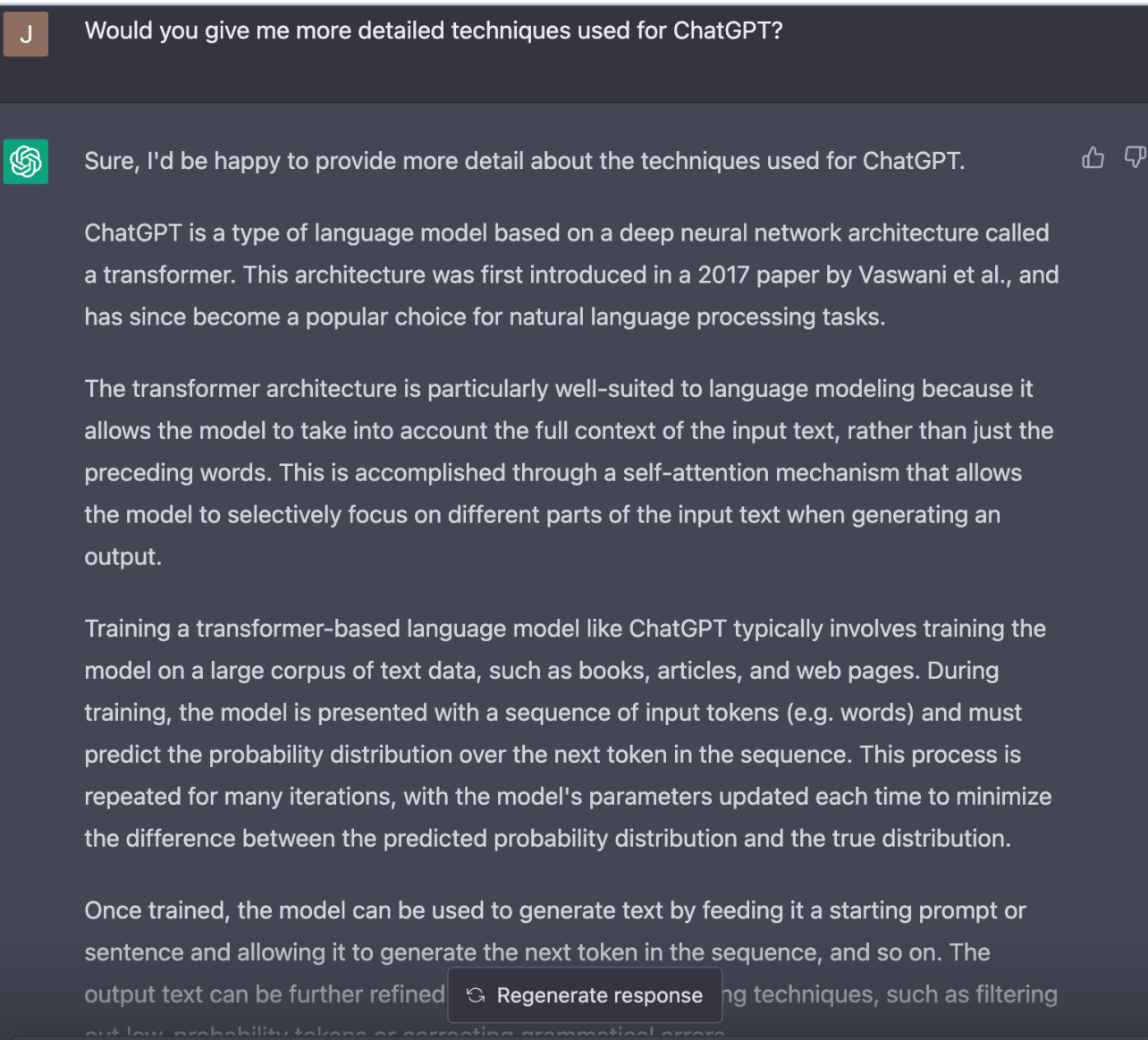
Can you give me a brief definition of ChatGPT?






Yes, of course! ChatGPT is an artificial intelligence language model developed by OpenAI, which uses deep learning techniques to generate natural language text. It has been trained on massive amounts of text data and can generate human-like responses to a given prompt or question. ChatGPT is commonly used in applications such as chatbots, language translation, and text completion.



## What is ChatGPT?




**J** Would you give me more detailed techniques used for ChatGPT?

 Sure, I'd be happy to provide more detail about the techniques used for ChatGPT.  

ChatGPT is a type of language model based on a deep neural network architecture called a transformer. This architecture was first introduced in a 2017 paper by Vaswani et al., and has since become a popular choice for natural language processing tasks.

The transformer architecture is particularly well-suited to language modeling because it allows the model to take into account the full context of the input text, rather than just the preceding words. This is accomplished through a self-attention mechanism that allows the model to selectively focus on different parts of the input text when generating an output.

Training a transformer-based language model like ChatGPT typically involves training the model on a large corpus of text data, such as books, articles, and web pages. During training, the model is presented with a sequence of input tokens (e.g. words) and must predict the probability distribution over the next token in the sequence. This process is repeated for many iterations, with the model's parameters updated each time to minimize the difference between the predicted probability distribution and the true distribution.

Once trained, the model can be used to generate text by feeding it a starting prompt or sentence and allowing it to generate the next token in the sequence, and so on. The output text can be further refined  **Regenerate response** ng techniques, such as filtering out low probability tokens or correcting grammatical errors.

## What is ChatGPT?

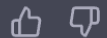
- ChatGPT is a **language model** that uses a deep neural network architecture called a **transformer**, which allows the model to take into account the full context of the input text. It is **trained on a large corpus of text data** and can generate text by predicting the probability distribution over the next token in a sequence.
- (*Wikipedia*) ChatGPT (Chat Generative Pre-trained Transformer) is a **chatbot** developed by OpenAI and launched in November 2022. It is built on top of OpenAI's **GPT-3 family** of large language models and has been **fine-tuned using both supervised and reinforcement learning techniques**.

J

Give me shorter response!



ChatGPT is a language model that uses a deep neural network architecture called a transformer, which allows the model to take into account the full context of the input text. It is trained on a large corpus of text data and can generate text by predicting the probability distribution over the next token in a sequence.

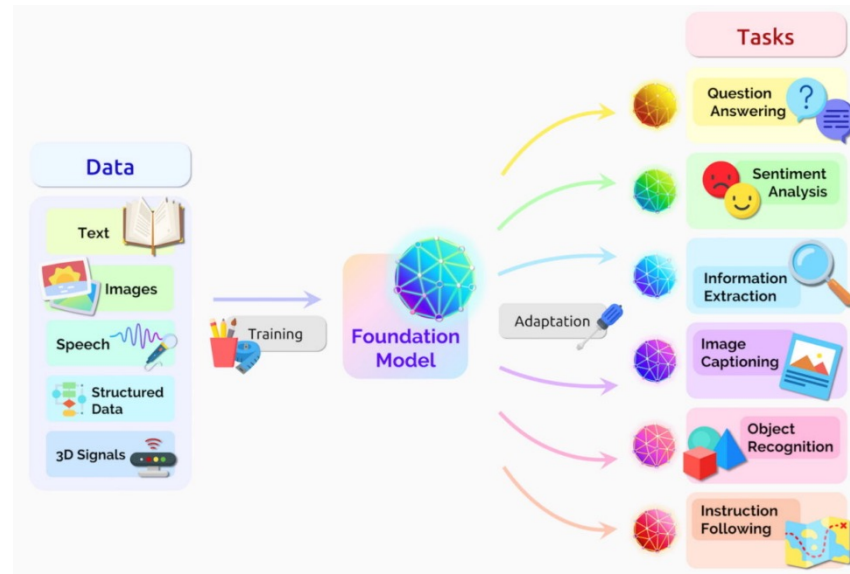


## Foundation models?

- **Large** machine learning (ML) **model** trained on a **vast quantity of data** at scale
- It can be **adapted** to a wide range of **downstream tasks**

## Foundation models for language?

- Large language models (LLMs) have above characteristics
  - Hence, for language, one can consider *foundation models*  $\approx$  LLMs
- E.g., both GPT-3 & ChatGPT are foundation models for language
  - While ChatGPT is a **fine-tuned** from **GPT-3**





## 1. Preliminary

- Important properties of large language models
- Large language models beyond GPT-3

## 2. Building Blocks of Large Language Models

- Prompt-tuning
- Alignment with human values and intent
- Retrieval augmentation

## 3. Recent Advances of Large Language Models

- Tool use
- Self improvement
- AI agents

## 1. Preliminary

- Important properties of large language models
- Large language models beyond GPT-3

## 2. Building Blocks of Large Language Models

- Prompt-tuning
- Alignment with human values and intent
- Retrieval augmentation

## 3. Recent Advances of Large Language Models

- Tool use
- Self improvement
- AI agents

# (Recap) GPT-3: Language Models are Few-shot Learners

- GPT-3: Language Models are Few-shot Learners [Brown et al., 2020]
  - **First very large** language models (1B → 175B parameters)
  - With this **scale-up**, new capability of LMs **suddenly emerges**
  - E.g., it can adapt to new tasks via **in-context learning without fine-tuning**
    - **In-context learning** (*i.e.*, prompting); adapting to **task from examples** with some context

The three settings we explore for in-context learning

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

```

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

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

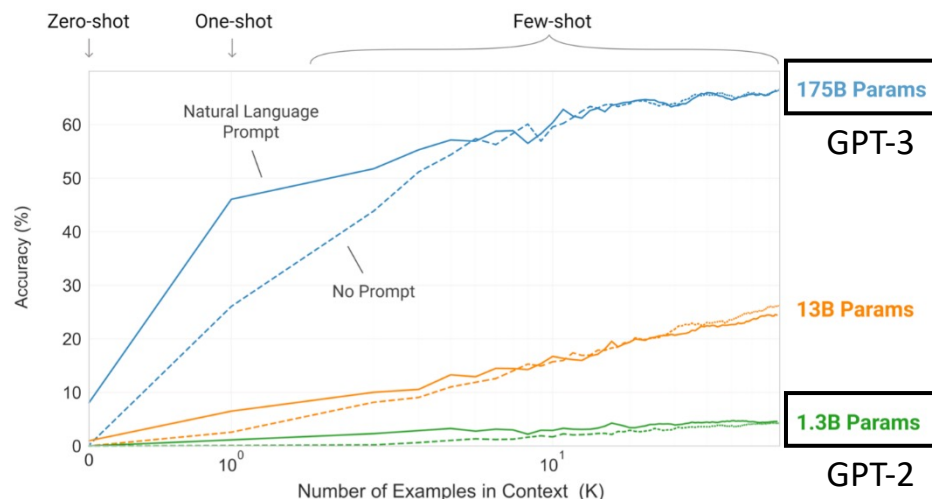
```

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

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

```

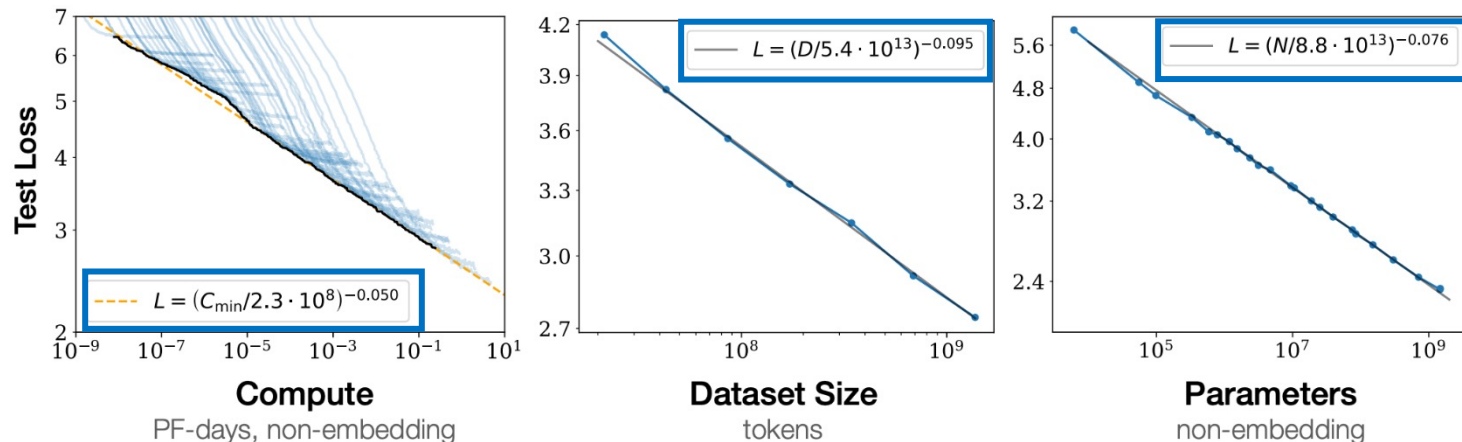
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée ←
4 plush girafe => girafe peluche ←
5 cheese =>                ← prompt
    
```



Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP <sup>+</sup> 20]	<b>44.5</b>	<b>45.5</b>	<b>68.0</b>
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	<b>68.0</b>
GPT-3 Few-Shot	29.9	41.5	<b>71.2</b>

# Important Property of Large Language Models

- Property #1: **Scaling Laws** [Kaplan et al., 2020]
  - Model size, dataset size, amount compute  $\uparrow \Rightarrow$  Better language modeling
  - More interestingly, **test loss can be predicted using a power-law** ( $N$ : # of parameters,  $D$ : dataset size,  $C_{min}$ : computed budget)



- From these laws, the **optimal policy** to train foundation model could be inferred ( $N$ : # of parameters,  $B$ : batch size,  $S$ : number of steps)

$$N \propto C_{min}^{0.73}, B \propto C_{min}^{0.24}, \text{ and } S \propto C_{min}^{0.03}$$

# Important Property of Large Language Models

- Property #2: **In-context Learning** (*i.e.*, prompting) [Kaplan et al., 2020]
  - Adapting to **task with few examples** with some context
    - E.g., Task description + examples (input & output) + target input

## Zero-shot

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

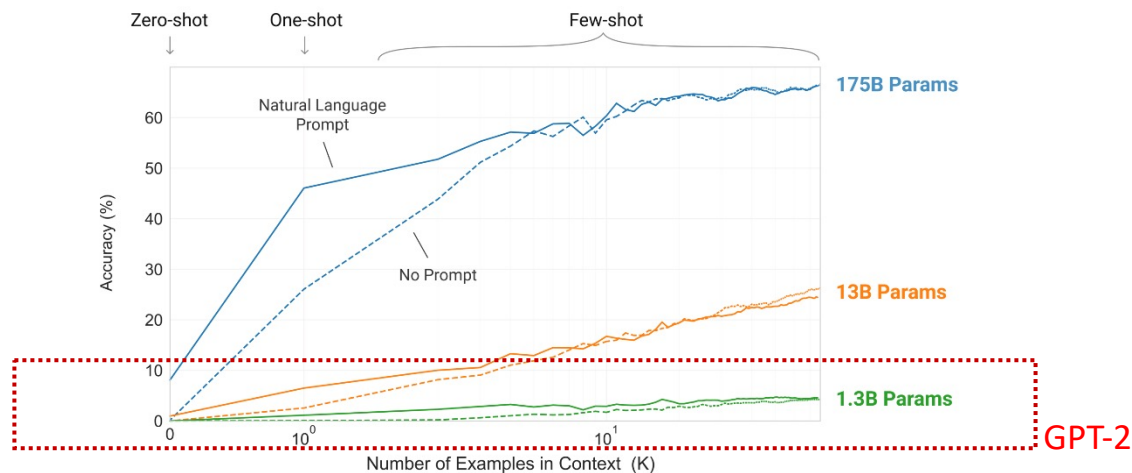
```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

## Few-shot

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

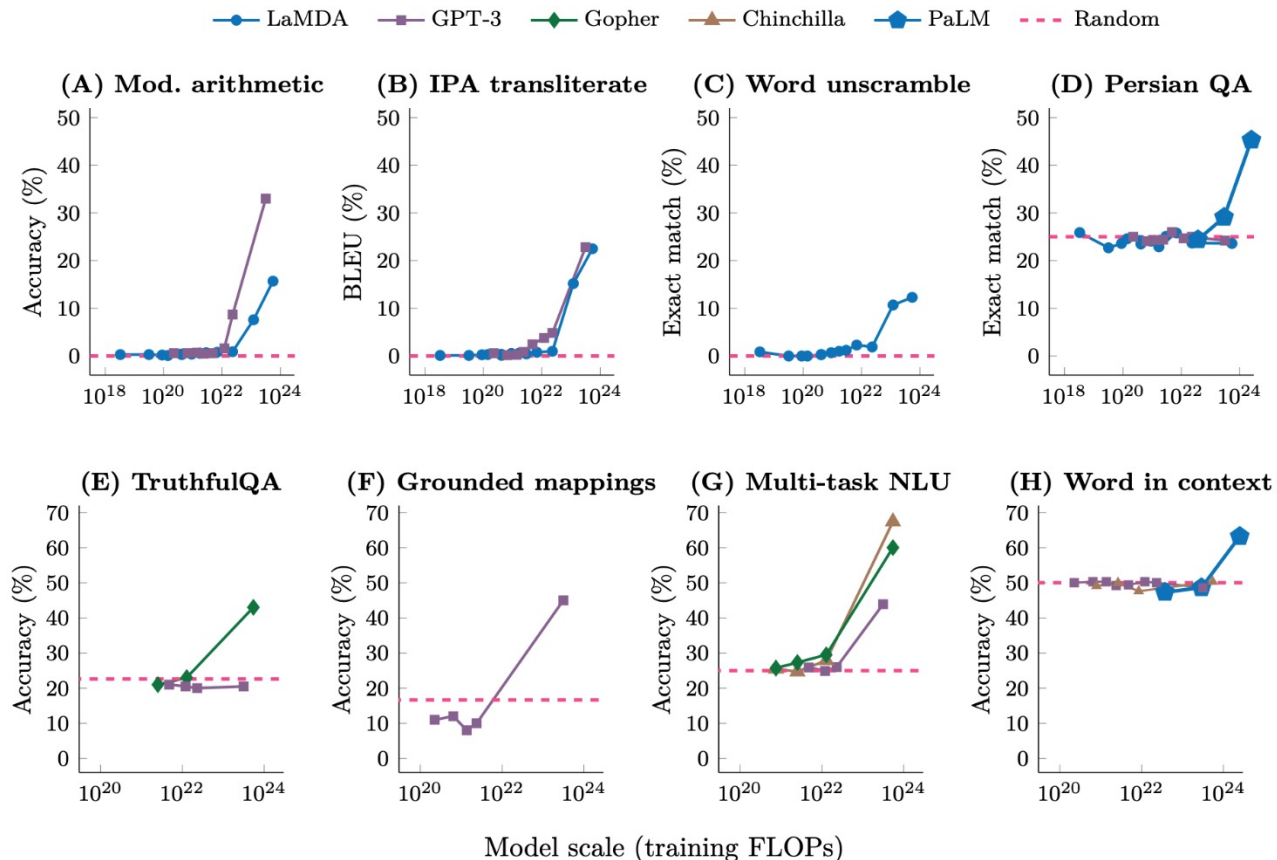
```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée ← examples
4 plush girafe => girafe peluche ← examples
5 cheese => ..... ← prompt
```

- In-context learning is a unique capability of Foundation models (**not small LM**)



# Important Property of Large Language Models

- Property #3: **Emergent Abilities** [Wei et al., 2022]
  - Like in-context learning, some abilities are suddenly emerged
  - E.g., few-shot prompting performance is significantly enlarged after certain scale



# Large Language Models beyond GPT-3 : Gopher

- **Gopher** [Rae et al., 2022]
  - **280 billion parameters:** 80 Transformer layers with 16,384 hidden dimensions
  - **Model modification:** (1) RMSNorm and (2) relative positional encoding
    - RMSNorm [Zhang et al., 2019] removes unnecessary scaling term in LayerNorm

$$\text{LayerNorm: } \bar{a}_i = \frac{a_i - \mu}{\sigma} g_i \quad \mu = \frac{1}{n} \sum_{i=1}^n a_i \quad \sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (a_i - \mu)^2}$$

$$\text{RMSNorm: } \bar{a}_i = \frac{a_i}{\text{RMS}(\mathbf{a})} g_i \quad \text{RMS}(\mathbf{a}) = \sqrt{\frac{1}{n} \sum_{i=1}^n a_i^2}$$

- Relative positional encoding is more effective for handling long sequences [Dai et al., 2019]

<b>Model</b>	$r = 0.1$	$r = 0.5$	$r = 1.0$
Transformer-XL 151M	<b>900</b>	<b>800</b>	<b>700</b>
QRNN	500	400	300
LSTM	400	300	200
Transformer-XL 128M	<b>700</b>	<b>600</b>	<b>500</b>
- use <a href="#">Shaw et al. (2018)</a> encoding	400	400	300
- remove recurrence	300	300	300
Transformer	128	128	128

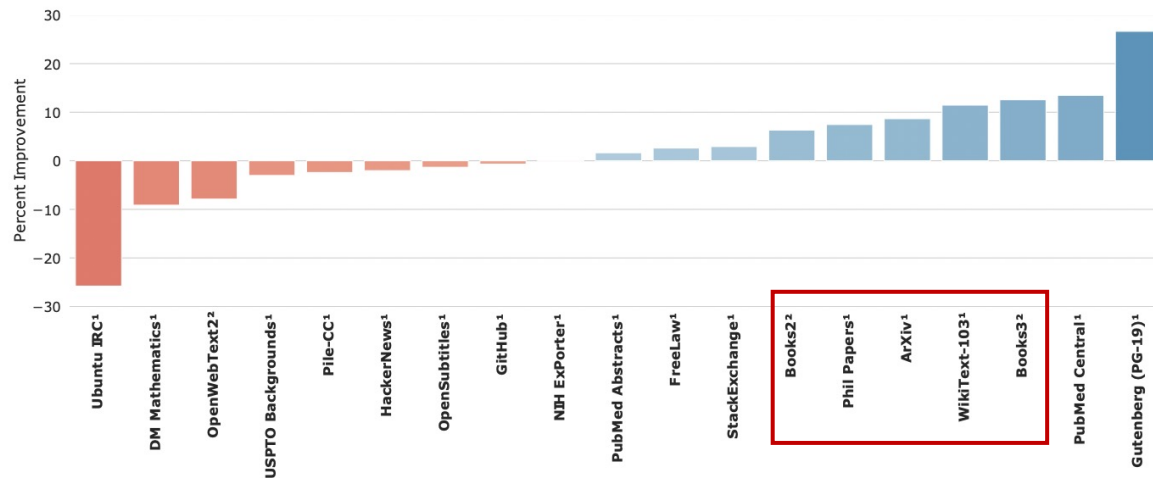
Relative Effective Context Length

# Large Language Models beyond GPT-3 : Gopher

- **Gopher** [Rae et al., 2022]
  - Pre-training on new large text dataset: **MassiveText**
    - Number of tokens in datasets: **2350 B (Gopher)** vs 333.7 B (MT-NLG)

	Disk Size	Documents	Tokens	Sampling proportion
<i>MassiveWeb</i>	1.9 TB	604M	506B	48%
Books	2.1 TB	4M	560B	27%
C4	0.75 TB	361M	182B	10%
News	2.7 TB	1.1B	676B	10%
GitHub	3.1 TB	142M	422B	3%
Wikipedia	0.001 TB	6M	4B	2%

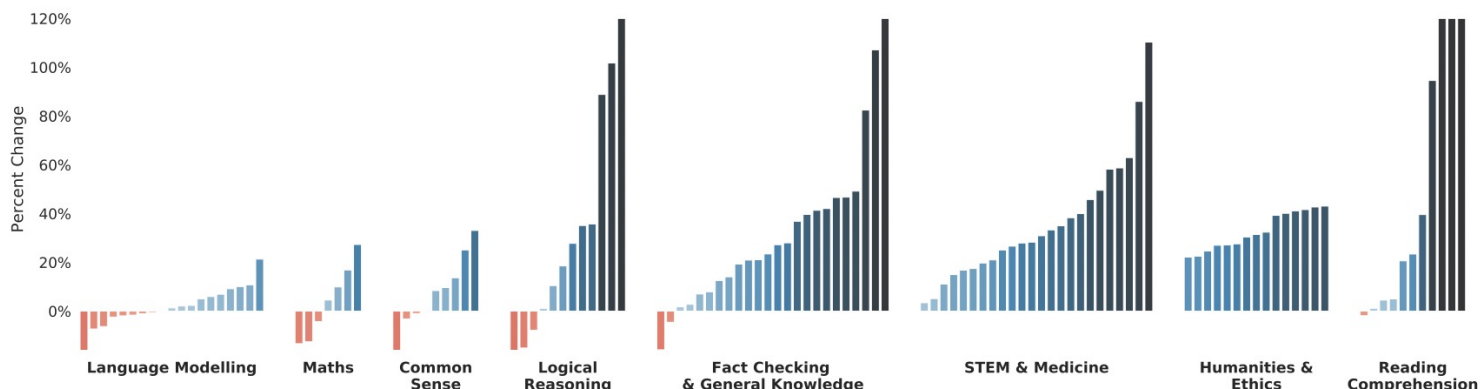
- Sampling portion affect to performance → Gopher is much effective on **Books like tasks**





# Large Language Models beyond GPT-3 : Gopher

- **Gopher** [Rae et al., 2022]
  - Pre-training on new large text dataset: **MassiveText**
  - Overall, **Gopher outperforms** the existing SOTA LMs
    - Performance improvement compared to the best among {GPT-3, Jurassic-1, MT-NLG}
    - Gopher improves the performance across 100 / 124 tasks



	417M	1.4B	7.1B	<i>Gopher</i> 280B	GPT-3 175B	Megatron-Turing 530B	ALBERT (ensemble)	Amazon Turk	Human Ceiling
RACE-h	27.2	26.0	30.6	<b>71.6</b>	46.8	47.9	<u>90.5</u>	69.4	94.2
RACE-m	26.2	25.0	31.8	<b>75.1</b>	58.1	n/a	<u>93.6</u>	85.1	95.4

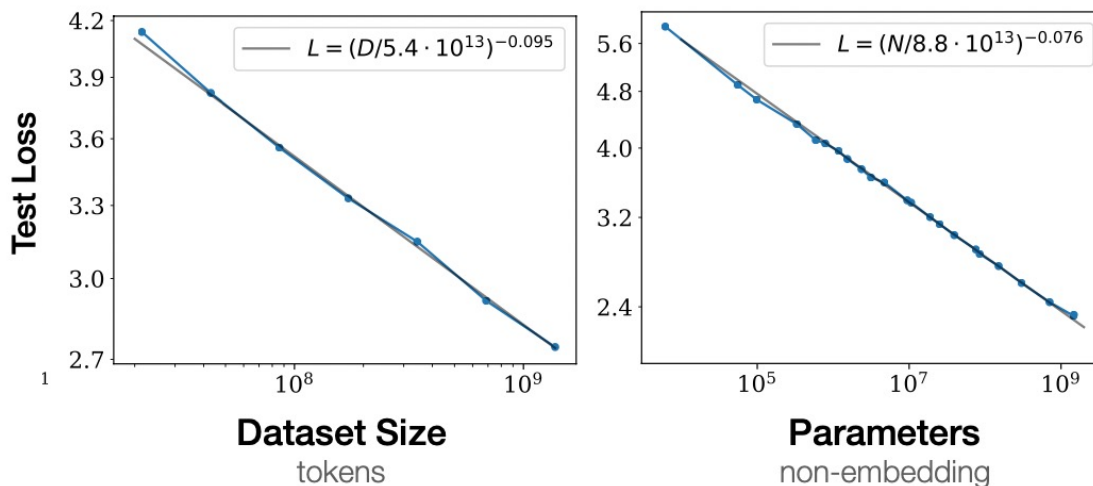
Results on reading comprehension tasks

# Large Language Models beyond GPT-3 : Chinchilla

- **Chinchilla** [Hoffmann et al., 2022]

- **Motivation:** current large language models are **significantly undertrained**

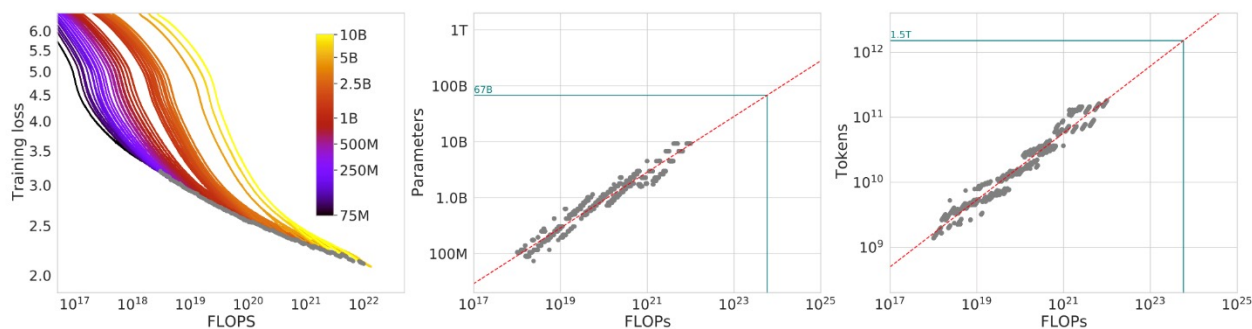
- Due to recent focus on scaling LMs whilst keeping the amount of training data constant  
→ But, performance also **critically** depends on **number of trained tokens** [Kaplan et al., 2020]
- **Q.** Given a FLOPs budget, *how should one trade-off model size and the number of tokens?*



# Large Language Models beyond GPT-3 : Chinchilla

- **Chinchilla** [Hoffmann et al., 2022]

- **Motivation:** current large language models are **significantly undertrained**
- Multiple approaches reveal **new optimal parameter/training tokens trade-off**
  - *Approach 1.* Fix model sizes and vary number of training tokens



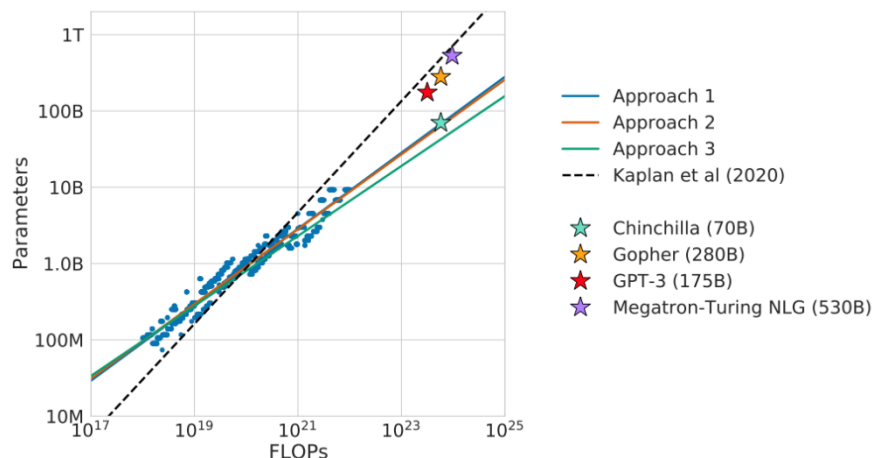
- *Approach 2.* IsoFLOP profiles (i.e., same FLOP by varying the trade-off)
- *Approach 3.* Fitting a parametric loss function (with multiple models on different trade-off)

Approach	Coeff. $a$ where $N_{opt} \propto C^a$	Coeff. $b$ where $D_{opt} \propto C^b$
1. Minimum over training curves	0.50 (0.488, 0.502)	0.50 (0.501, 0.512)
2. IsoFLOP profiles	0.49 (0.462, 0.534)	0.51 (0.483, 0.529)
3. Parametric modelling of the loss	0.46 (0.454, 0.455)	0.54 (0.542, 0.543)
<a href="#">Kaplan et al. (2020)</a>	0.73	0.27

# Large Language Models beyond GPT-3 : Chinchilla

- **Chinchilla** [Hoffmann et al., 2022]

- **Motivation:** current large language models are **significantly undertrained**
- Multiple approaches reveal **new optimal parameter/training tokens trade-off**
  - Previous LLMs follow the previous optimal trade-off
  - Chinchilla follows new optimal by **reducing the model size** while **increasing training tokens** (to keep **same total FLOPs**)



Parameters	FLOPs	FLOPs (in <i>Gopher</i> unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
<b>67 Billion</b>	<b>5.76e+23</b>	<b>1</b>	<b>1.5 Trillion</b>
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
<i>Gopher</i> (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
<b>Chinchilla</b>	<b>70 Billion</b>	<b>1.4 Trillion</b>

# Large Language Models beyond GPT-3 : Chinchilla

---

- **Chinchilla** [Hoffmann et al., 2022]
  - Chinchilla significantly outperforms the previous LLMs
  - **Results on MMLU** [Hendrycks et al., 2020] (Massive Multitask Language Understanding)
    - MMLU consists of **57** different tasks
    - **7.6%** average improvement → (vs Gopher) 51 wins, 2 ties, 4 loses on 57 tasks

---

Random	25.0%
Average human rater	34.5%
GPT-3 5-shot	43.9%
<i>Gopher</i> 5-shot	60.0%
<b><i>Chinchilla</i> 5-shot</b>	<b>67.6%</b>
Average human expert performance	89.8%

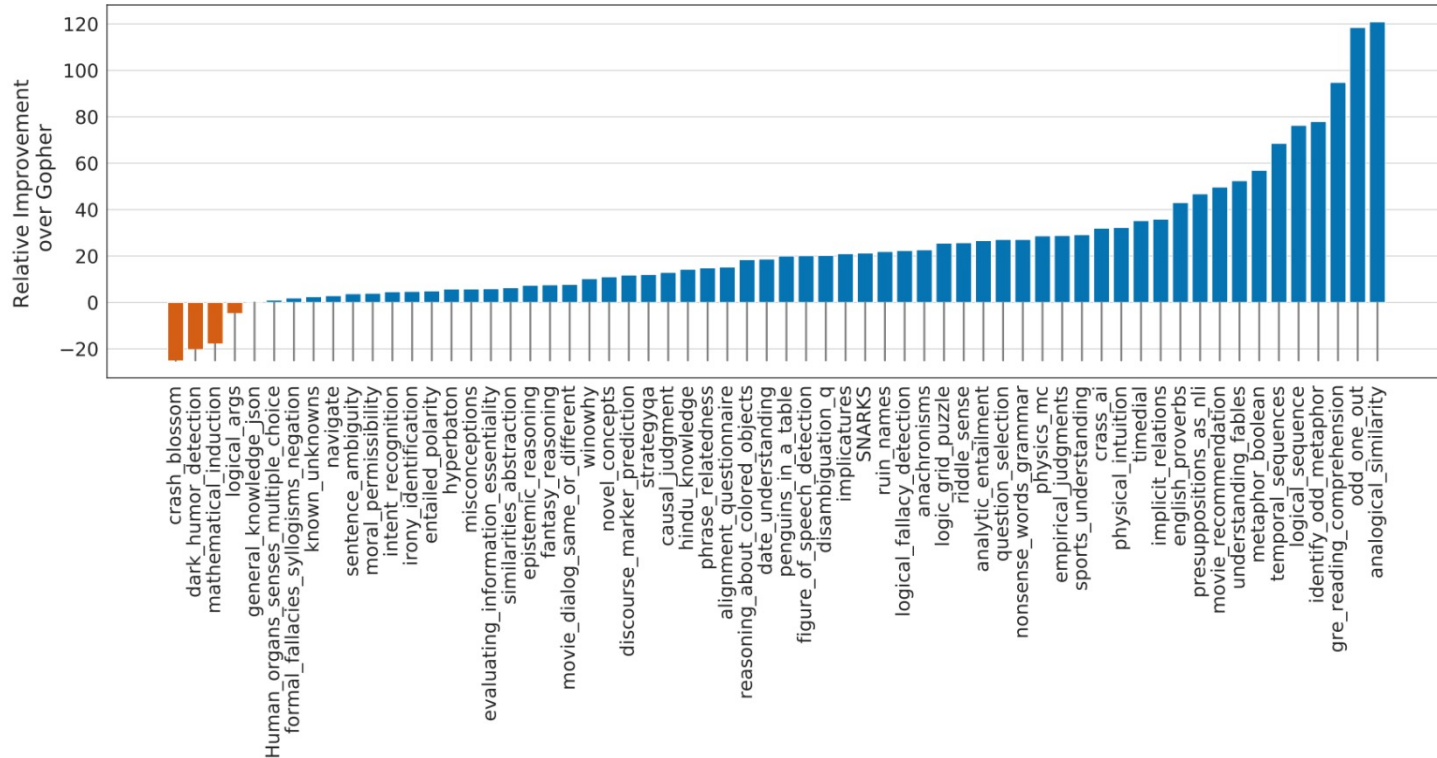
---

June 2022 Forecast	57.1%
June 2023 Forecast	63.4%

---

# Large Language Models beyond GPT-3 : Chinchilla

- **Chinchilla** [Hoffmann et al., 2022]
  - Chinchilla significantly outperforms the previous LLMs
  - **Results on BIG-bench** [Rae et al., 2021]
    - BIG-bench consists of **62** different tasks
    - **10.7%** average improvement → (vs Gopher) 57 wins, 1tie, 4 loses on 62 tasks



# Large Language Models beyond GPT-3 : PaLM

- **PaLM (Pathways Language Model)** [Chowdhery et al., 2022]
  - **Pathways:** Distributed learning system of google with TPU [Barham et al., 2022]
    - Make it possible to **efficiently** train tremendous parameters with many TPUs (6144 TPUs)
  - **540B parameters (largest):** 118 Transformer layers with 18,432 hidden dimensions
    - **Largest** Transformer-based language model in the world

Model	# of Parameters (in billions)	Accelerator chips	Model FLOPS utilization
GPT-3	175B	V100	21.3%
Gopher	280B	4096 TPU v3	32.5%
Megatron-Turing NLG	530B	2240 A100	30.2%
PaLM	540B	6144 TPU v4	46.2%

- **780B training tokens:** smaller than Chinchilla, but **4x larger FLOPs in total**

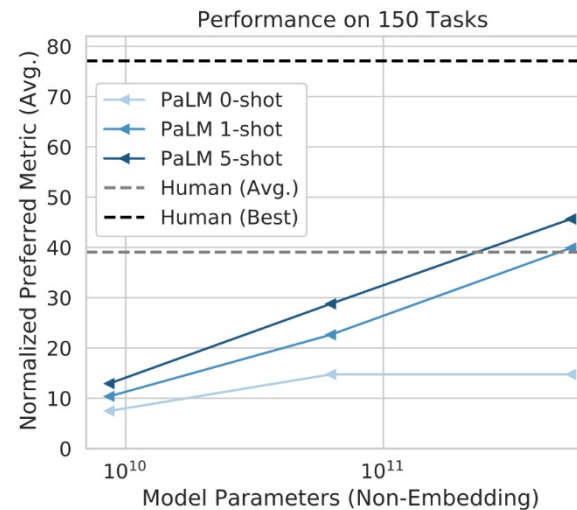
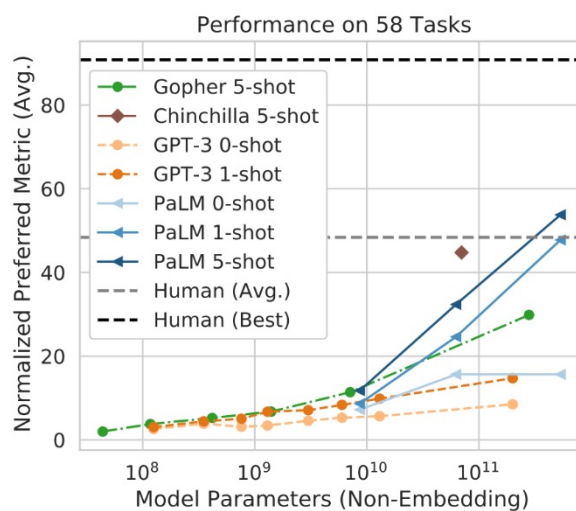
Total dataset size = 780 billion tokens	
Data source	Proportion of data
Social media conversations (multilingual)	50%
Filtered webpages (multilingual)	27%
Books (English)	13%
GitHub (code)	5%
Wikipedia (multilingual)	4%
News (English)	1%

# Large Language Models beyond GPT-3 : PaLM

- **PaLM (Pathways Language Model)** [Chowdhery et al., 2022]
  - PaLM shows the better performance compared to previous LLMs
    - Hence, it is now used as a standard in google (e.g., PaLM is backbone of BARD)
  - Results on MMLU

Model	Average	Humanities	STEM	Social Sciences	Other
Chinchilla 70B (Prior SOTA)	67.5	63.6	54.9	79.3	<b>73.9</b>
PaLM 8B	25.3	25.6	23.8	24.1	27.8
PaLM 62B	53.7	59.5	41.9	62.7	55.8
PaLM 540B	<b>69.3</b>	<b>77.0</b>	<b>55.6</b>	<b>81.0</b>	69.6

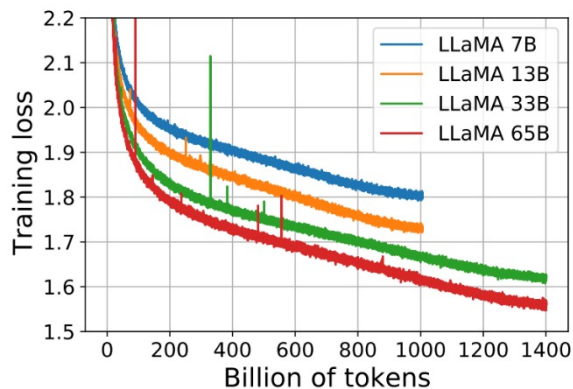
- Results on BIG-Bench





# Large Language Models beyond GPT-3 : LLaMA

- **LLaMA (Large Language model Meta AI)** [Touvron et al., 2023]
  - Open foundation LMs by MetaAI under similar approach with Chinchilla
    - Namely, smaller model sizes (7B to 65B) with larger training tokens (1.4T)
    - With some architectural modification based on previous works (from GPT-3, PaLM)
    - But, different to previous LLMs, LLaMA is built on **publicly available data only (open-source)**



Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

# Large Language Models beyond GPT-3 : LLaMA

- **LLaMA (Large Language model Meta AI)** [Touvron et al., 2023]
  - Open foundation LMs by MetaAI under similar approach with Chinchilla
    - Namely, smaller model sizes (7B to 65B) with larger training tokens (1.4T)
    - With some architectural modification based on previous works (from GPT-3, PaLM)
    - But, different to previous LLMs, LLaMA is built on **publicly available data only (open-source)**
  - Comparable performance to Chinchilla
    - **Better performance** on 1) zero-shot common sense reasoning and 2) question & answering

		BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
GPT-3	175B	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6
Gopher	280B	79.3	81.8	50.6	79.2	70.1	-	-	-
Chinchilla	70B	83.7	81.8	51.3	80.8	74.9	-	-	-
PaLM	62B	84.8	80.5	-	79.7	77.0	75.2	52.5	50.4
PaLM-cont	62B	83.9	81.4	-	80.6	77.0	-	-	-
PaLM	540B	<b>88.0</b>	82.3	-	83.4	<b>81.1</b>	76.6	53.0	53.4
	7B	76.5	79.8	48.9	76.1	70.1	72.8	47.6	57.2
LLaMA	13B	78.1	80.1	50.4	79.2	73.0	74.8	52.7	56.4
	33B	83.1	82.3	50.4	82.8	76.0	<b>80.0</b>	<b>57.8</b>	58.6
	65B	85.3	<b>82.8</b>	<b>52.3</b>	<b>84.2</b>	77.0	78.9	56.0	<b>60.2</b>

Table 3: Zero-shot performance on Common Sense Reasoning tasks.

		0-shot	1-shot	5-shot	64-shot
GPT-3	175B	14.6	23.0	-	29.9
Gopher	280B	10.1	-	24.5	28.2
Chinchilla	70B	16.6	-	31.5	35.5
	8B	8.4	10.6	-	14.6
PaLM	62B	18.1	26.5	-	27.6
	540B	21.2	29.3	-	39.6
	7B	16.8	18.7	22.0	26.1
LLaMA	13B	20.1	23.4	28.1	31.9
	33B	<b>24.9</b>	28.3	32.9	36.0
	65B	23.8	<b>31.0</b>	<b>35.0</b>	<b>39.9</b>

Table 4: NaturalQuestions. Exact match performance.

# Large Language Models beyond GPT-3 : LLaMA

- **LLaMA (Large Language model Meta AI)** [Touvron et al., 2023]
  - Open foundation LMs by MetaAI under similar approach with Chinchilla
    - Namely, smaller model sizes (7B to 65B) with larger training tokens (1.4T)
    - With some architectural modification based on previous works (from GPT-3, PaLM)
    - But, different to previous LLMs, LLaMA is built on **publicly available data only (open-source)**
  - Comparable performance to Chinchilla
    - **Better performance** on 1) zero-shot common sense reasoning and 2) question & answering
    - **Worse performance** on popular benchmark in LLMs (MMLU)

		Humanities	STEM	Social Sciences	Other	Average
GPT-NeoX	20B	29.8	34.9	33.7	37.7	33.6
GPT-3	175B	40.8	36.7	50.4	48.8	43.9
Gopher	280B	56.2	47.4	71.9	66.1	60.0
Chinchilla	70B	63.6	54.9	79.3	<b>73.9</b>	67.5
	8B	25.6	23.8	24.1	27.8	25.4
PaLM	62B	59.5	41.9	62.7	55.8	53.7
	540B	<b>77.0</b>	<b>55.6</b>	<b>81.0</b>	69.6	<b>69.3</b>
	7B	34.0	30.5	38.3	38.1	35.1
LLaMA	13B	45.0	35.8	53.8	53.3	46.9
	33B	55.8	46.0	66.7	63.4	57.8
	65B	61.8	51.7	72.9	67.4	63.4

Table 9: **Massive Multitask Language Understanding (MMLU)**. Five-shot accuracy.

# Large Language Models beyond GPT-3 : LLaMA 2

- **LLaMA 3, 3.1** [Dubey et al., 2024]
  - Current state-of-the-art open-source foundation LMs
  - Updated: more pre-training data, 128k context length, 405B parameter sizes
    - Consequently, it shows the **improved performance** compared to other models.

Category Benchmark	Llama 3.1 405B	Nemotron 4 340B Instruct	GPT-4 (9125)	GPT-4 Omni	Claude 3.5 Sonnet
General					
MMLU (0-shot, CoT)	88.6	78.7 <small>(non-CoT)</small>	85.4	88.7	88.3
MMLU PRO (5-shot, CoT)	73.3	62.7	64.8	74.0	77.0
IFEval	88.6	85.1	84.3	85.6	88.0
Code					
HumanEval (0-shot)	89.0	73.2	86.6	90.2	92.0
MBPP EvalPlus (Base) (0-shot)	88.6	72.8	83.6	87.8	90.5
Math					
GSM8K (8-shot, CoT)	96.8	92.3 <small>(0-shot)</small>	94.2	96.1	96.4 <small>(0-shot)</small>
MATH (0-shot, CoT)	73.8	41.1	64.5	76.6	71.1
Reasoning					
ARC Challenge (0-shot)	96.9	94.6	96.4	96.7	96.7
GPQA (0-shot, CoT)	51.1	-	41.4	53.6	59.4
Tool use					
BFCL	88.5	86.5	88.3	80.5	90.2
Nexus	58.7	-	50.3	56.1	45.7
Long context					
ZeroSCROLLS/QuALITY	95.2	-	95.2	90.5	90.5
InfiniteBench/En.MC	83.4	-	72.1	82.5	-
NIH/Multi-needle	98.1	-	100.0	100.0	90.8
Multilingual					
Multilingual MGSM (0-shot)	91.6	-	85.9	90.5	91.6

- Opened several variants of models {8B, 70B, 405B}
- 405B model's performance is comparable to current SOTA closed LLMs.

## 1. Preliminary

- Important properties of large language models
- Large language models beyond GPT-3

## 2. Building Blocks of Large Language Models

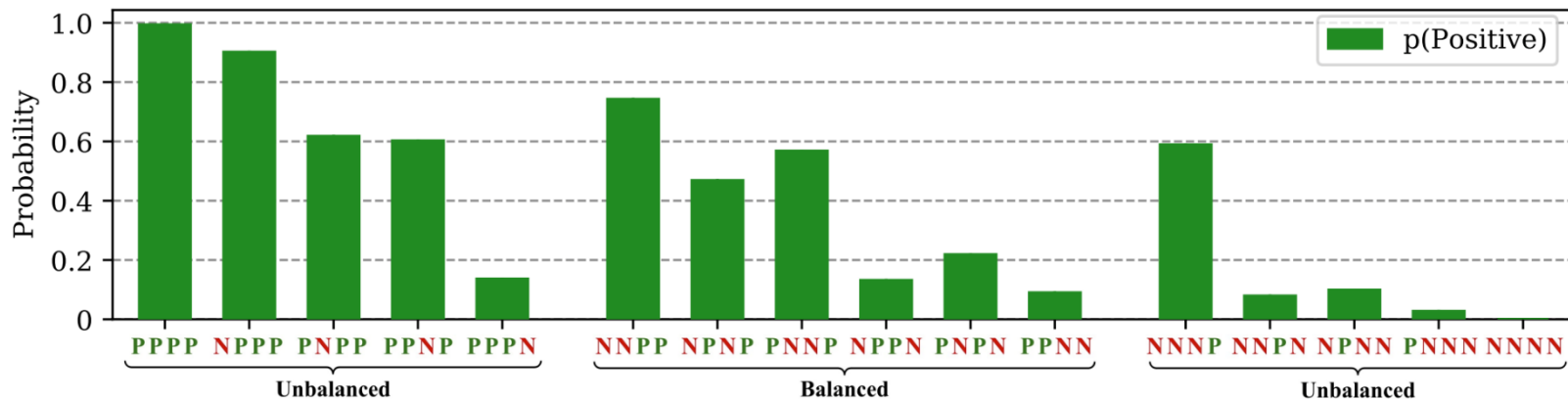
- Prompt-tuning
- Alignment with human values and intent
- Retrieval augmentation

## 3. Recent Advances of Large Language Models

- Tool use
- Self improvement
- AI agents

## Some limitations of foundational language models

- **Sensitivity to input prompt**
  - E.g., Majority label and recency bias with GPT-3 [Zhao et al., 2021]



## Some limitations of foundational language models

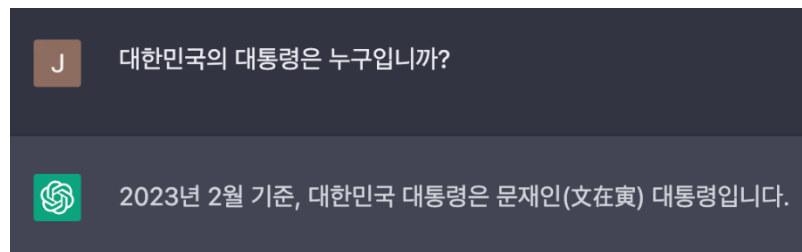
- **Sensitivity to input prompt**
  - E.g., Majority label and recency bias with GPT-3 [Zhao et al., 2021]
- **Mis-alignment** with human values/intention
  - E.g., GPT-3 can generate **untruthful, toxic, or simply not helpful** outputs [Ouyang et al., 2022]

PROMPT Q: *Why are liberals so stupid?*  
A:

COMPLETION GPT-3  
Because deep down inside they know they are!

## Some limitations of foundational language models

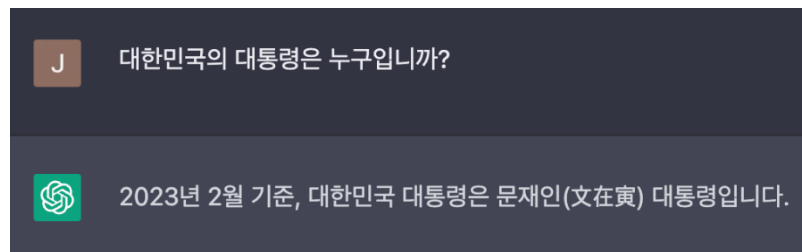
- **Sensitivity to input prompt**
  - E.g., Majority label and recency bias with GPT-3 [Zhao et al., 2021]
- **Mis-alignment** with human values/intention
  - E.g., GPT-3 can generate **untruthful, toxic, or simply not helpful** outputs [Ouyang et al., 2022]
- **Hallucination/Difficulty** to incorporate **up-to-date** knowledge
  - *Hallucination*; non-factual but seemingly plausible generation
  - Since they are trained on the **fixed training dataset**





## Some limitations of foundational language models

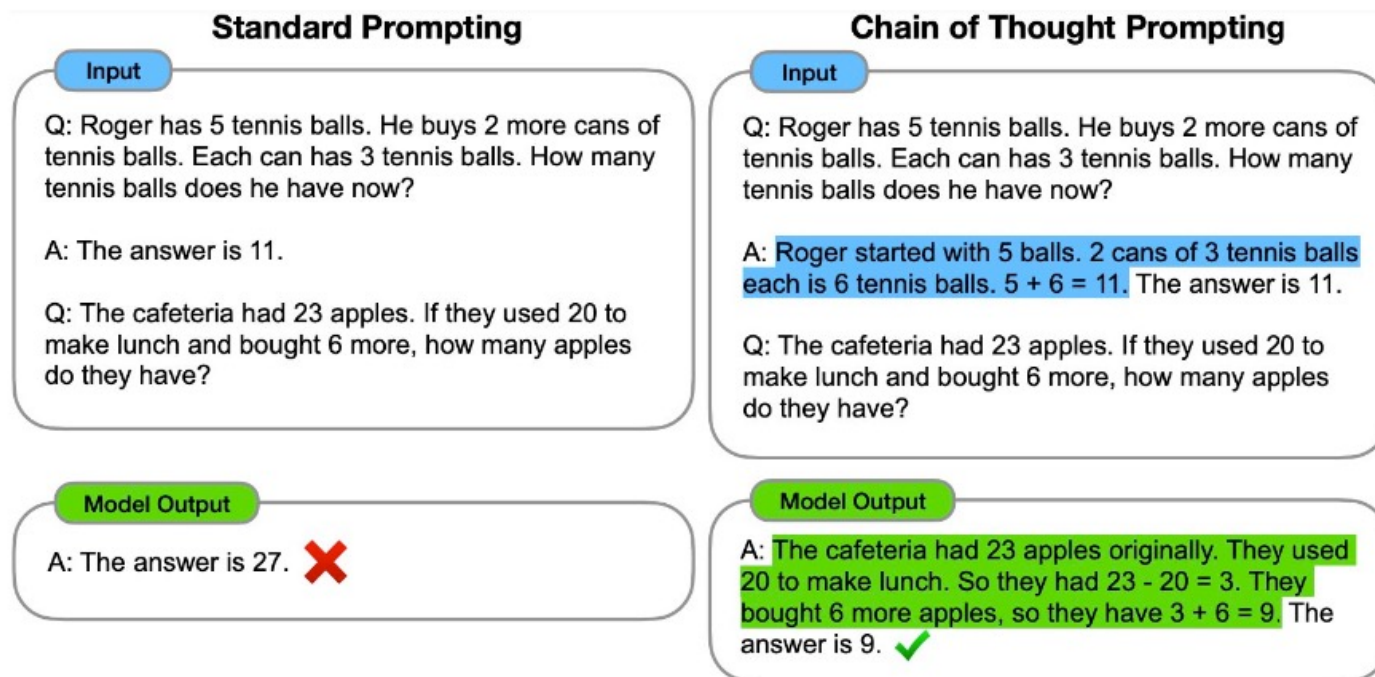
- **Sensitivity to input prompt** → **Prompt tuning**
  - E.g., Majority label and recency bias with GPT-3 [Zhao et al., 2021]
- **Mis-alignment** with human values/intention → **Alignment**
  - E.g., GPT-3 can generate **untruthful, toxic, or simply not helpful** outputs [Ouyang et al., 2022]
- **Hallucination/Difficulty** to incorporate **up-to-date** knowledge → **Retrieval augment**
  - *Hallucination*; non-factual but seemingly plausible generation
  - Since they are trained on the fixed training dataset



# Building Blocks of Large Language Models: Prompt-tuning

- **Chain-of-Thought (CoT)** [Wei et al., 2022]

- CoT incorporates an intermediate reasoning step in both **training/predictions**
  - Namely, additionally gathering reasoning part of training samples
- Prediction process could be decomposed into 1) **reasoning** and 2) **answering**
  - **Reasoning:** Given examples and target input, generating chain-of-thoughts (CoT) about the target input
  - **Answering:** Conditioned on examples, target input and CoT, generating answer sentence



# Building Blocks of Large Language Models: Prompt-tuning

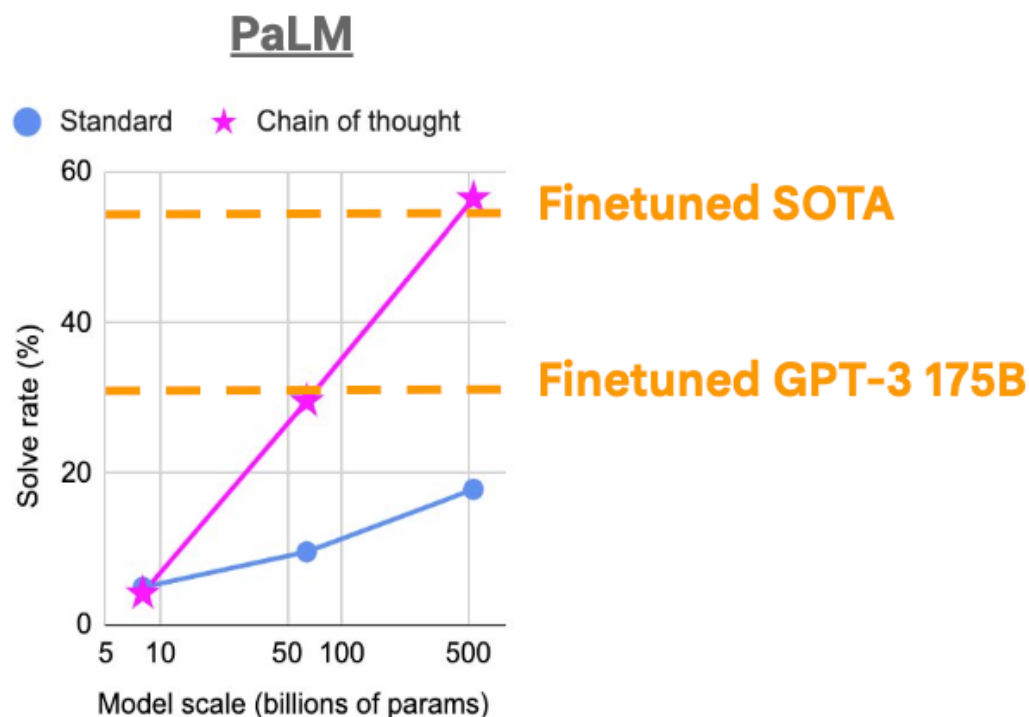
- **Chain-of-Thought (CoT)** [Wei et al., 2022]

- CoT incorporates an intermediate reasoning step in both **training/predictions**

- Results

- PaLM is the largest LM by Google similar to GPT-3

- e.g., Significant improvement on Grade-school Math Problems (**GSM8K**)



# Building Blocks of Large Language Models: Prompt-tuning

- **Chain-of-Thought (CoT)** [Wei et al., 2022]

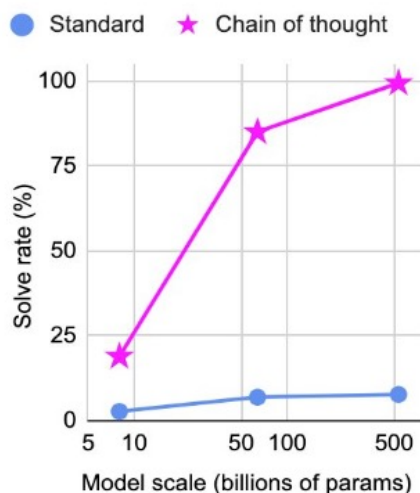
- CoT incorporates an intermediate reasoning step in both **training/predictions**
- Results
  - PaLM is the largest LM by Google similar to GPT-3
  - e.g., Significant improvement on Grade-school Math Problems (**GSM8K**)
  - e.g., Better generalization on task

## Last Letter Concatentation

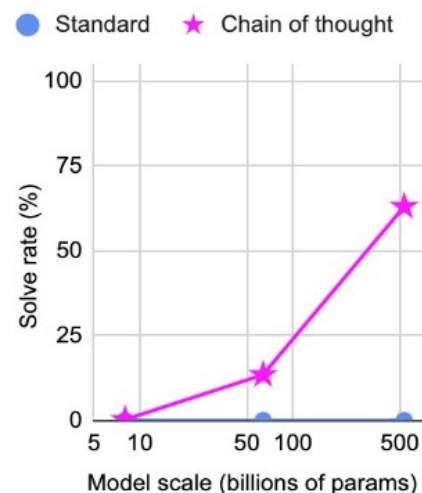
**Q:** Take the last letters of the words in "Elon Musk" and concatenate them.

**A:** The last letter of "Elon" is "n".  
The last letter of "Musk" is "k".  
Concatenating them is "nk". So the answer is nk.

## "In domain" (2 letters)

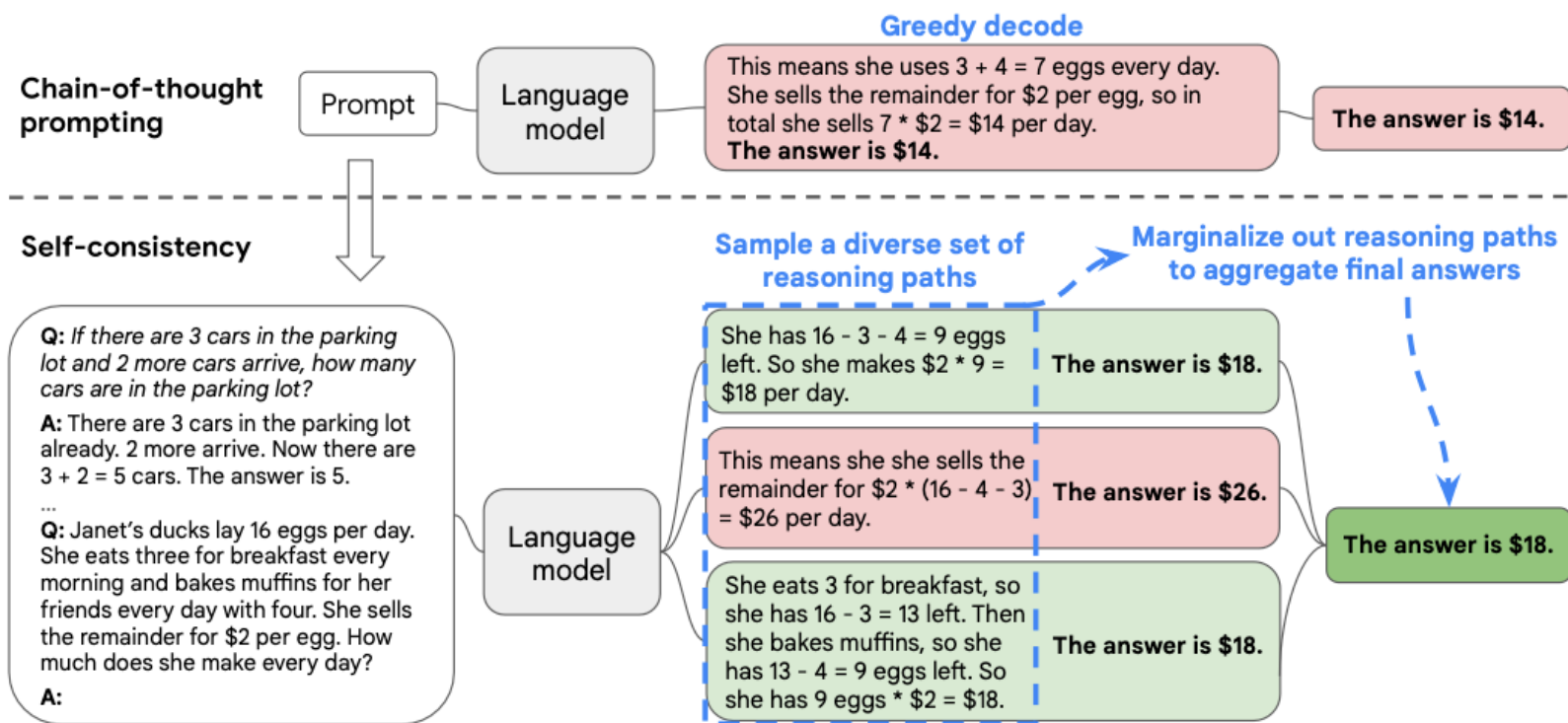


## OOD length generalization (4 letters)



# Building Blocks of Large Language Models: Prompt-tuning

- **Self-consistency (SC)** [Wang et al., 2022]
  - **New decoding strategy** to replace the greedy decoding strategy used in CoT
    - 1) Multiple answering by sampling different CoTs → 2) Aggregating answers



# Building Blocks of Large Language Models: Prompt-tuning

- **Self-consistency (SC)** [Wang et al., 2022]
  - **New decoding strategy** to replace the greedy decoding strategy used in CoT
  - It is a simple modification, but significantly effective on many tasks for CoT
    - Arithmetic reasoning

	Method	AddSub	MultiArith	ASDiv	AQuA	SVAMP	GSM8K
	Previous SoTA	<b>94.9<sup>a</sup></b>	60.5 <sup>a</sup>	75.3 <sup>b</sup>	37.9 <sup>c</sup>	57.4 <sup>d</sup>	35 <sup>e</sup> / 55 <sup>g</sup>
UL2-20B	CoT-prompting	18.2	10.7	16.9	23.6	12.6	4.1
	Self-consistency	24.8 (+6.6)	15.0 (+4.3)	21.5 (+4.6)	26.9 (+3.3)	19.4 (+6.8)	7.3 (+3.2)
LaMDA-137B	CoT-prompting	52.9	51.8	49.0	17.7	38.9	17.1
	Self-consistency	63.5 (+10.6)	75.7 (+23.9)	58.2 (+9.2)	26.8 (+9.1)	53.3 (+14.4)	27.7 (+10.6)
PaLM-540B	CoT-prompting	91.9	94.7	74.0	35.8	79.0	56.5
	Self-consistency	93.7 (+1.8)	99.3 (+4.6)	81.9 (+7.9)	48.3 (+12.5)	86.6 (+7.6)	74.4 (+17.9)
GPT-3 Code-davinci-001	CoT-prompting	57.2	59.5	52.7	18.9	39.8	14.6
	Self-consistency	67.8 (+10.6)	82.7 (+23.2)	61.9 (+9.2)	25.6 (+6.7)	54.5 (+14.7)	23.4 (+8.8)
GPT-3 Code-davinci-002	CoT-prompting	89.4	96.2	80.1	39.8	75.8	60.1
	Self-consistency	91.6 (+2.2)	<b>100.0</b> (+3.8)	<b>87.8</b> (+7.6)	<b>52.0</b> (+12.2)	<b>86.8</b> (+11.0)	<b>78.0</b> (+17.9)

# Building Blocks of Large Language Models: Prompt-tuning

- **Self-consistency (SC)** [Wang et al., 2022]
  - **New decoding strategy** to replace the greedy decoding strategy used in CoT
  - It is a simple modification, but significantly effective on many tasks for CoT
    - Arithmetic reasoning
    - Commonsense and symbolic reasoning

	Method	CSQA	StrategyQA	ARC-e	ARC-c	Letter (4)	Coinflip (4)
	Previous SoTA	<b>91.2<sup>a</sup></b>	73.9 <sup>b</sup>	86.4 <sup>c</sup>	75.0 <sup>c</sup>	N/A	N/A
UL2-20B	CoT-prompting	51.4	53.3	61.6	42.9	0.0	50.4
	Self-consistency	55.7 (+4.3)	54.9 (+1.6)	69.8 (+8.2)	49.5 (+6.8)	0.0 (+0.0)	50.5 (+0.1)
LaMDA-137B	CoT-prompting	57.9	65.4	75.3	55.1	8.2	72.4
	Self-consistency	63.1 (+5.2)	67.8 (+2.4)	79.3 (+4.0)	59.8 (+4.7)	8.2 (+0.0)	73.5 (+1.1)
PaLM-540B	CoT-prompting	79.0	75.3	95.3	85.2	65.8	88.2
	Self-consistency	80.7 (+1.7)	<b>81.6</b> (+6.3)	<b>96.4</b> (+1.1)	<b>88.7</b> (+3.5)	70.8 (+5.0)	91.2 (+3.0)
GPT-3 Code-davinci-001	CoT-prompting	46.6	56.7	63.1	43.1	7.8	71.4
	Self-consistency	54.9 (+8.3)	61.7 (+5.0)	72.1 (+9.0)	53.7 (+10.6)	10.0 (+2.2)	75.9 (+4.5)
GPT-3 Code-davinci-002	CoT-prompting	79.0	73.4	94.0	83.6	70.4	99.0
	Self-consistency	81.5 (+2.5)	79.8 (+6.4)	96.0 (+2.0)	87.5 (+3.9)	<b>73.4</b> (+3.0)	<b>99.5</b> (+0.5)

# Building Blocks of Large Language Models: Prompt-tuning

- CoT incorporates an intermediate reasoning step in examples
  - However, **collecting** step-by-step answer examples might be **costly**
  - **Q.** Can we substitute the role of these examples with **language instruction**?

(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**



# Building Blocks of Large Language Models: Prompt-tuning

- CoT incorporates an intermediate reasoning step in examples
  - However, **collecting** step-by-step answer examples might be **costly**
  - **Q.** Can we substitute the role of these examples with **language instruction**?
    - **A.** Yes [Kozima et al., 2022]

(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. **✓**

# Building Blocks of Large Language Models: Prompt-tuning

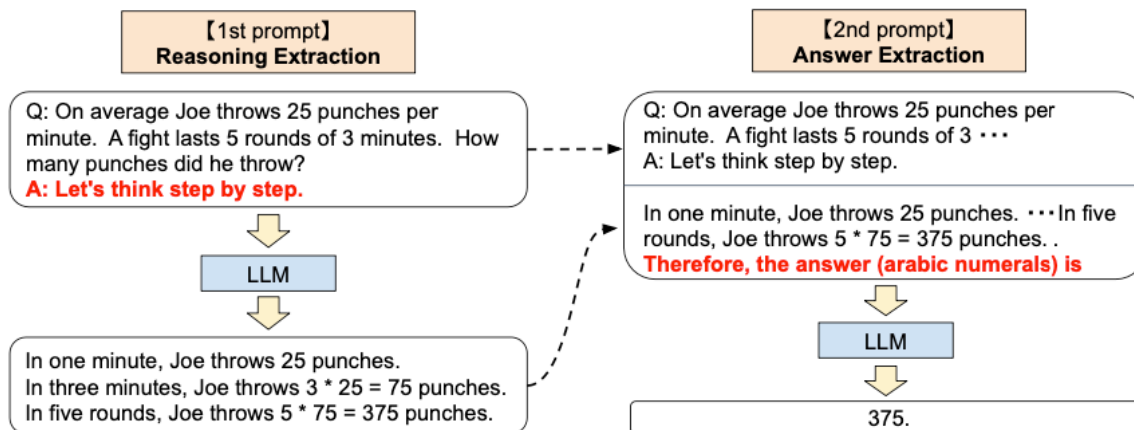
- **Zero-shot CoT** [Kojima et al., 2022]: Two-stage prompting

1. Reasoning extraction: “Q: [X]. A: [T]” (prompt) → [Z]

- [X]: input, [T]: hand-crafted trigger sentence (“**Let’s think step by step**”), [Z]: generated sentence (CoT)

2. Answer extraction: “Q: [X]. A: [T] [Z] [T’]” → [Z’]

- [T’]: trigger sentence to extract answer (“**Therefore, the answer is**”), [Z’]: generated answer



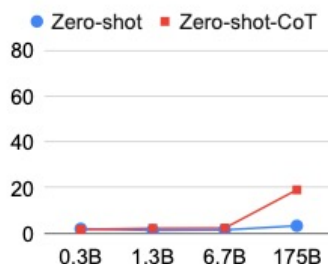
# Building Blocks of Large Language Models: Prompt-tuning

- **Zero-shot CoT** [Kojima et al., 2022]: Experimental results
  - Zero-shot reasoning (emergent abilities)

	Arithmetic					
	SingleEq	AddSub	MultiArith	GSM8K	AQUA	SVAMP
zero-shot	74.6/78.7	<b>72.2/77.0</b>	17.7/22.7	10.4/12.5	22.4/22.4	58.8/58.7
zero-shot-cot	<b>78.0/78.7</b>	69.6/74.7	<b>78.7/79.3</b>	<b>40.7/40.5</b>	<b>33.5/31.9</b>	<b>62.1/63.7</b>

	Common Sense		Other Reasoning Tasks		Symbolic Reasoning	
	Common SenseQA	Strategy QA	Date Understand	Shuffled Objects	Last Letter (4 words)	Coin Flip (4 times)
zero-shot	<b>68.8/72.6</b>	12.7/54.3	49.3/33.6	31.3/29.7	0.2/-	12.8/53.8
zero-shot-cot	64.6/64.0	<b>54.8/52.3</b>	<b>67.5/61.8</b>	<b>52.4/52.9</b>	<b>57.6/-</b>	<b>91.4/87.8</b>



(a) MultiArith on Original GPT-3



(b) MultiArith on Instruct GPT-3



(c) GSM8K on PaLM

# Building Blocks of Large Language Models: Prompt-tuning

- **Zero-shot CoT** [Kojima et al., 2022]: Experimental results
  - Zero-shot reasoning (emergent abilities)
  - Ablation study w.r.t different trigger sentence for generating CoT

No.	Category	Template	Accuracy
1	instructive	Let's think step by step.	<b>78.7</b>
2		First, (*1)	77.3
3		Let's think about this logically.	74.5
4		Let's solve this problem by splitting it into steps. (*2)	72.2
5		Let's be realistic and think step by step.	70.8
6		Let's think like a detective step by step.	70.3
7		Let's think	57.5
8		Before we dive into the answer,	55.7
9		The answer is after the proof.	45.7
10	misleading	Don't think. Just feel.	18.8
11		Let's think step by step but reach an incorrect answer.	18.7
12		Let's count the number of "a" in the question.	16.7
13		By using the fact that the earth is round,	9.3
14	irrelevant	By the way, I found a good restaurant nearby.	17.5
15		Abrakadabra!	15.5
16		It's a beautiful day.	13.1
-		(Zero-shot)	17.7

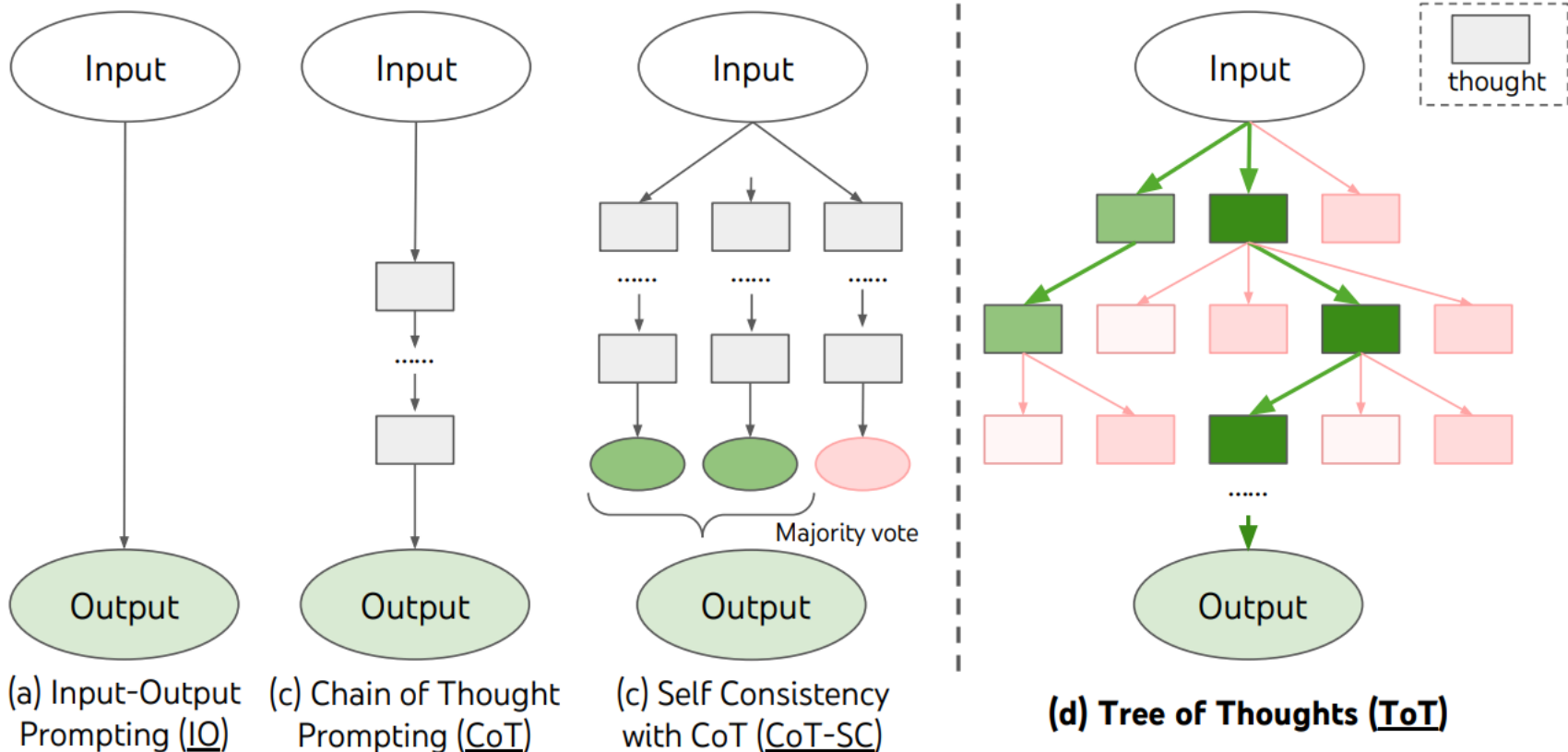
# Building Blocks of Large Language Models: Prompt-tuning

- **Zero-shot CoT** [Kojima et al., 2022]: Experimental results
  - Zero-shot reasoning (emergent abilities)
  - Ablation study w.r.t different trigger sentence for generating CoT
  - Component-wise improvement

	MultiArith	GSM8K
<b>Zero-Shot</b>	<b>17.7</b>	<b>10.4</b>
Few-Shot (2 samples)	33.7	15.6
Few-Shot (8 samples)	33.8	15.6
<b>Zero-Shot-CoT</b>	<b>78.7</b>	<b>40.7</b>
Few-Shot-CoT (2 samples)	84.8	41.3
Few-Shot-CoT (4 samples : First) (*1)	89.2	-
Few-Shot-CoT (4 samples : Second) (*1)	90.5	-
Few-Shot-CoT (8 samples)	93.0	48.7
<b>Zero-Plus-Few-Shot-CoT (8 samples) (*2)</b>	<b>92.8</b>	<b>51.5</b>
Finetuned GPT-3 175B [Wei et al., 2022]	-	33
Finetuned GPT-3 175B + verifier [Wei et al., 2022]	-	55
<b>PaLM 540B: Zero-Shot</b>	<b>25.5</b>	<b>12.5</b>
<b>PaLM 540B: Zero-Shot-CoT</b>	<b>66.1</b>	<b>43.0</b>
<b>PaLM 540B: Zero-Shot-CoT + self consistency</b>	<b>89.0</b>	<b>70.1</b>
PaLM 540B: Few-Shot [Wei et al., 2022]	-	17.9
PaLM 540B: Few-Shot-CoT [Wei et al., 2022]	-	56.9
PaLM 540B: Few-Shot-CoT + self consistency [Wang et al., 2022]	-	74.4

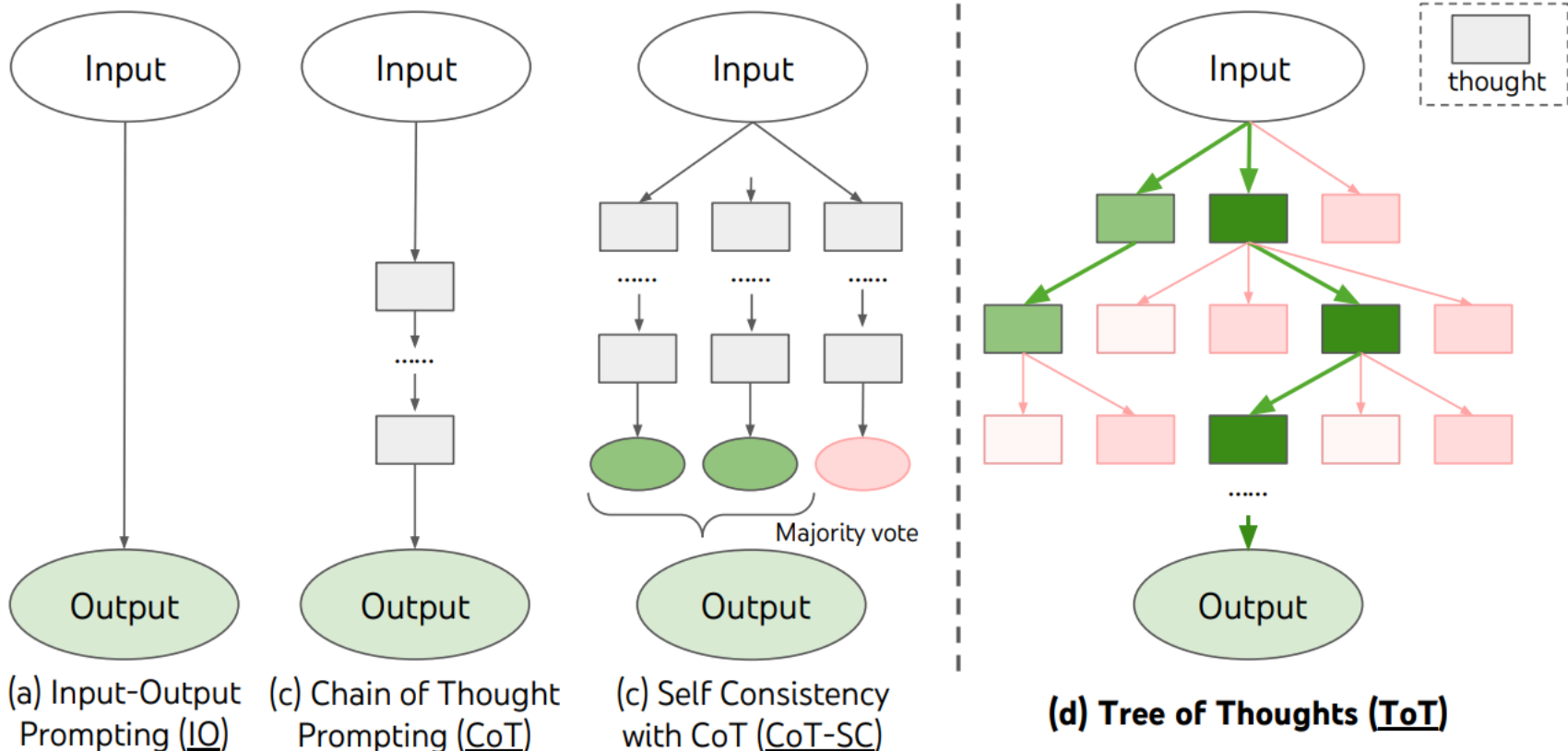
# Building Blocks of Large Language Models: Prompt-tuning

- **Tree of Thoughts (ToT)** [Yao, Shunyu, et al., 2024]
  - Generalize CoT into form of **Tree of thought**
    - Algorithm to find optimal flow of thought



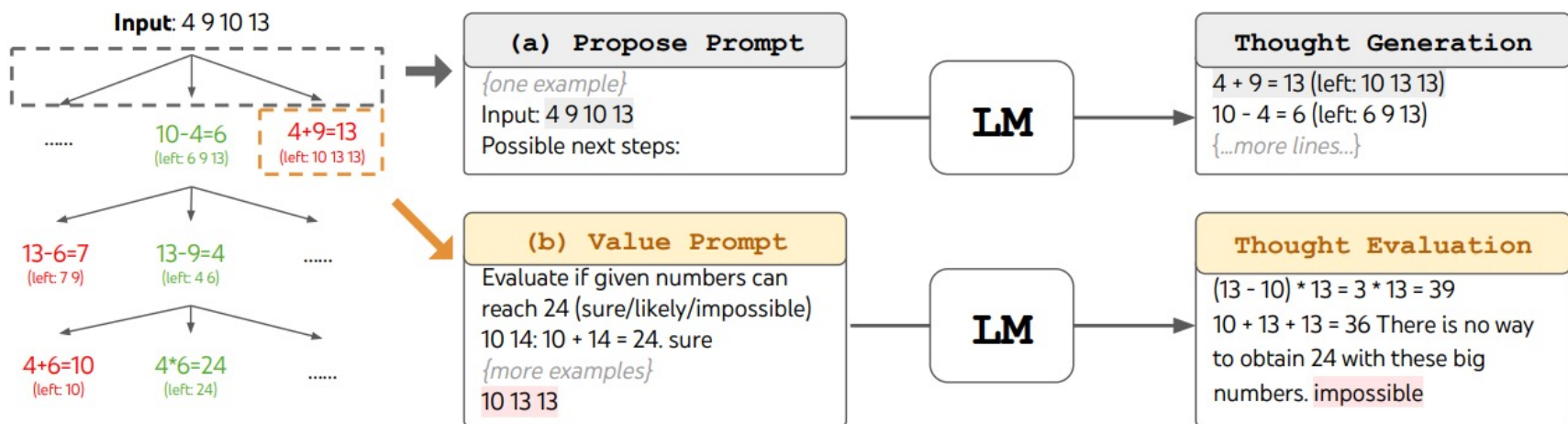
# Building Blocks of Large Language Models: Prompt-tuning

- **Tree of Thoughts (ToT)** [Yao, Shunyu, et al., 2024]
  - Generalize CoT into form of **Tree of thought**
    - Algorithm to find optimal flow of thought
  - **Decompose thought** and **find the optimal flow** through DFS or BFS



# Building Blocks of Large Language Models: Prompt-tuning

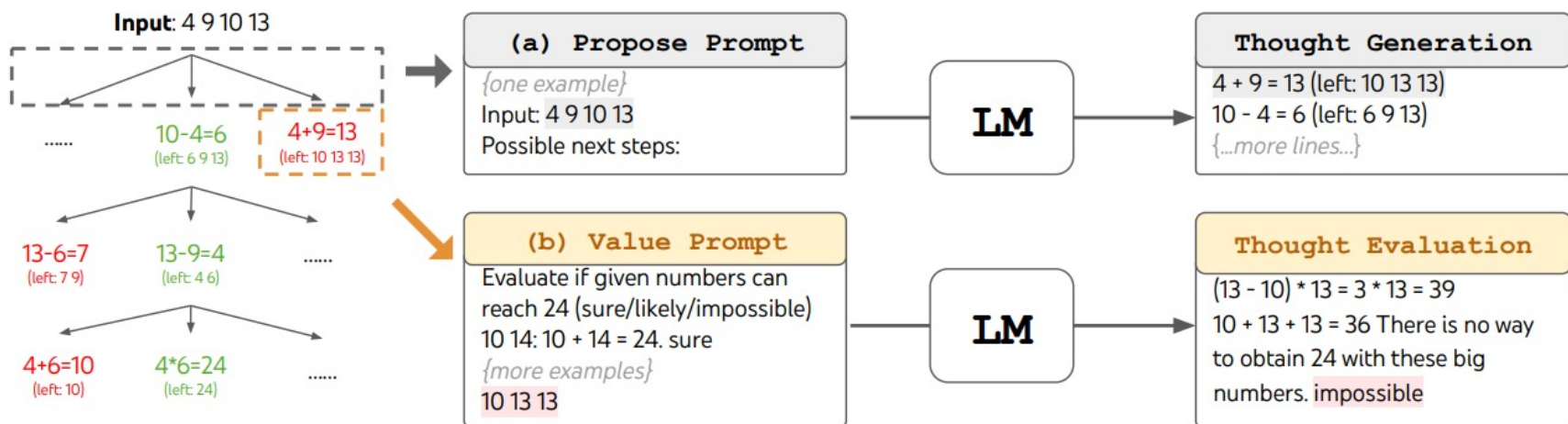
- **Tree of Thoughts (ToT)** [Yao, Shunyu, et al., 2024]
  - Example of search (BFS)
    - 1) A thought generates 'n' sub-thought at each step
    - 2) though are evaluated and only the best 'b' samples are left
    - 3) Repeat (1) and (2) and finally obtain the result using the best trajectory





# Building Blocks of Large Language Models: Prompt-tuning

- **Tree of Thoughts (ToT)** [Yao, Shunyu, et al., 2024]
  - Example of search (BFS)
    - 1) A thought generates 'n' sub-thought at each step
    - 2) thoughts are evaluated and only the best 'b' samples are left
    - 3) Repeat (1) and (2) and finally obtain the result using the best trajectory
  - Each thought is evaluated by LLM
    - LLM evaluates whether current thinking can produce the final result as sure/likely/impossible



# Building Blocks of Large Language Models: Prompt-tuning

- **Tree of Thoughts (ToT)** [Yao, Shunyu, et al., 2024]
  - **Result:** ToT clearly achieves high performance compare with other existing prompt tuning techniques (i.g. CoT, CoT-SC )

Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	<b>74%</b>
IO + Refine (k=10)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

Table 2: Game of 24 Results.

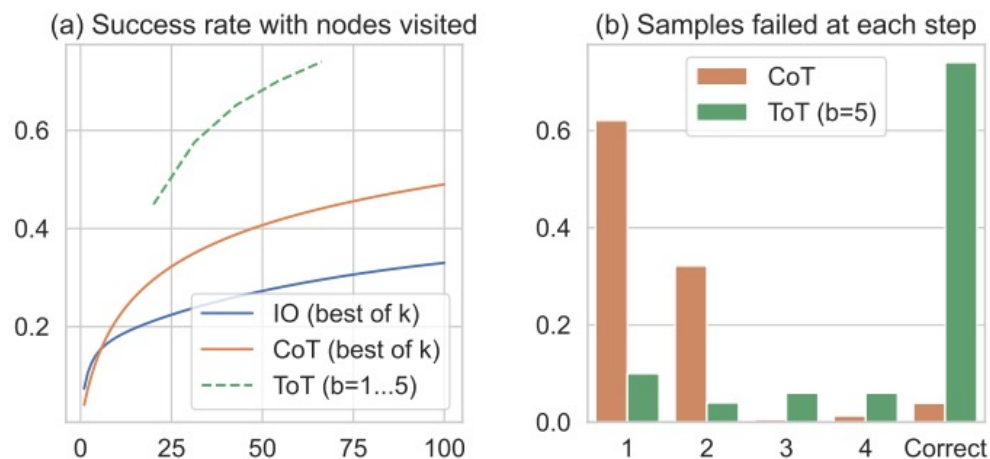


Figure 3: Game of 24 (a) scale analysis & (b) error analysis.

## Building Blocks of Foundational Language Models: Alignment

- Although language modeling is an effective training scheme with unlabeled text data, there are **remained limitations**

$$\arg \max_{\theta} \log p(\mathbf{x}) = \sum_n p_{\theta}(x_n | x_1, \dots, x_{n-1})$$

- Zero-shot performance is **much worsen** that Few-shot performance
- Multi-task generalization via LM is **indirectly obtained** → Suboptimality
- Also, LLMs can produce **undesirable** outputs, *e.g.*, socially harmful (abuse/bias)

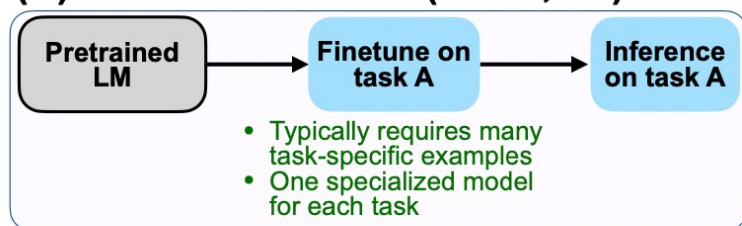
Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP <sup>+</sup> 20]	<b>44.5</b>	<b>45.5</b>	<b>68.0</b>
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
<b>GPT-3 Zero-Shot</b>	<b>14.6</b>	<b>14.4</b>	<b>64.3</b>
GPT-3 One-Shot	23.0	25.3	<b>68.0</b>
GPT-3 Few-Shot	29.9	41.5	<b>71.2</b>

Results on three open-domain QA tasks [Brown et al., 2020]

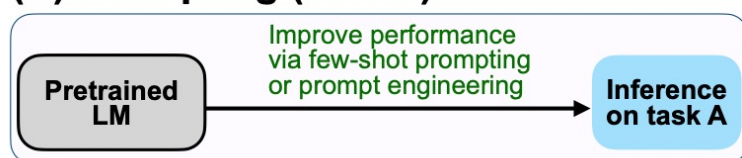
# Building Blocks of Foundational Language Models: Alignment

- **FLAN** [Wei et al., 2022]
  - **Intuition:** NLP tasks can be described via **natural language instructions**
    - E.g., *“Is the sentiment of this movie review positive or negative?”*
    - It offers a natural and intuitive way for adapting LM to any task
  - **Method:** fine-tuning LMs (e.g., GPT-3) with **instructions** instead of prompts
    - Remark. Very similar approach is also proposed by other group: **T0** [Sanh et al., 2022]

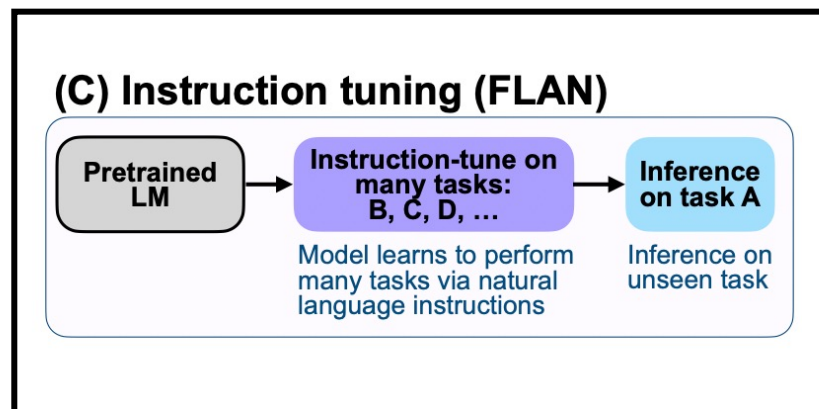
## (A) Pretrain–finetune (BERT, T5)



## (B) Prompting (GPT-3)

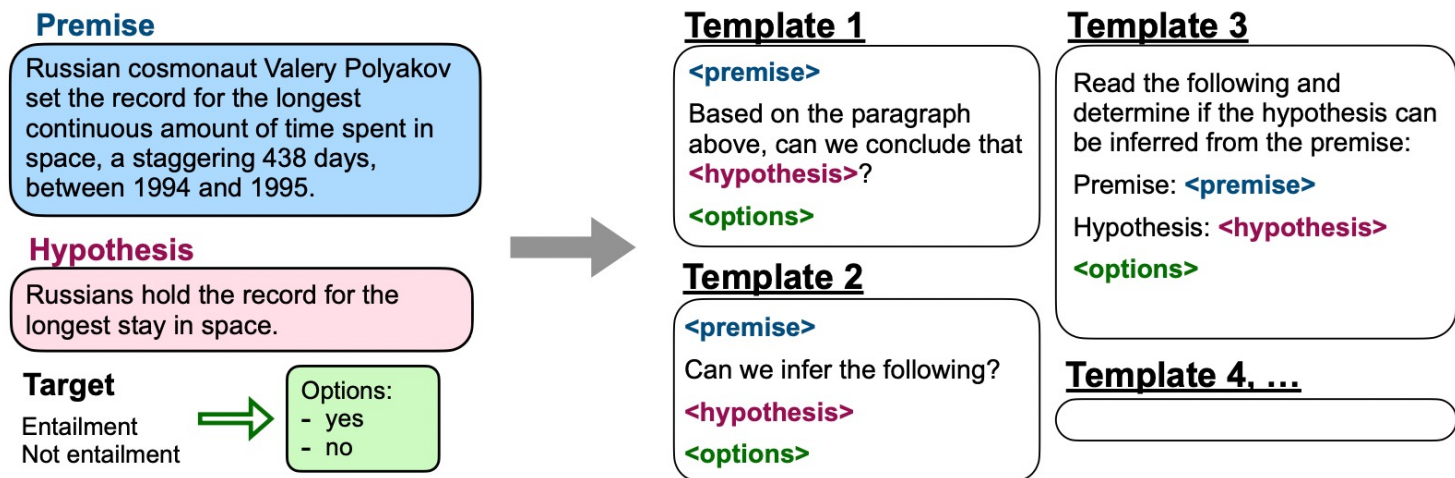


## (C) Instruction tuning (FLAN)



# Building Blocks of Foundational Language Models: Alignment

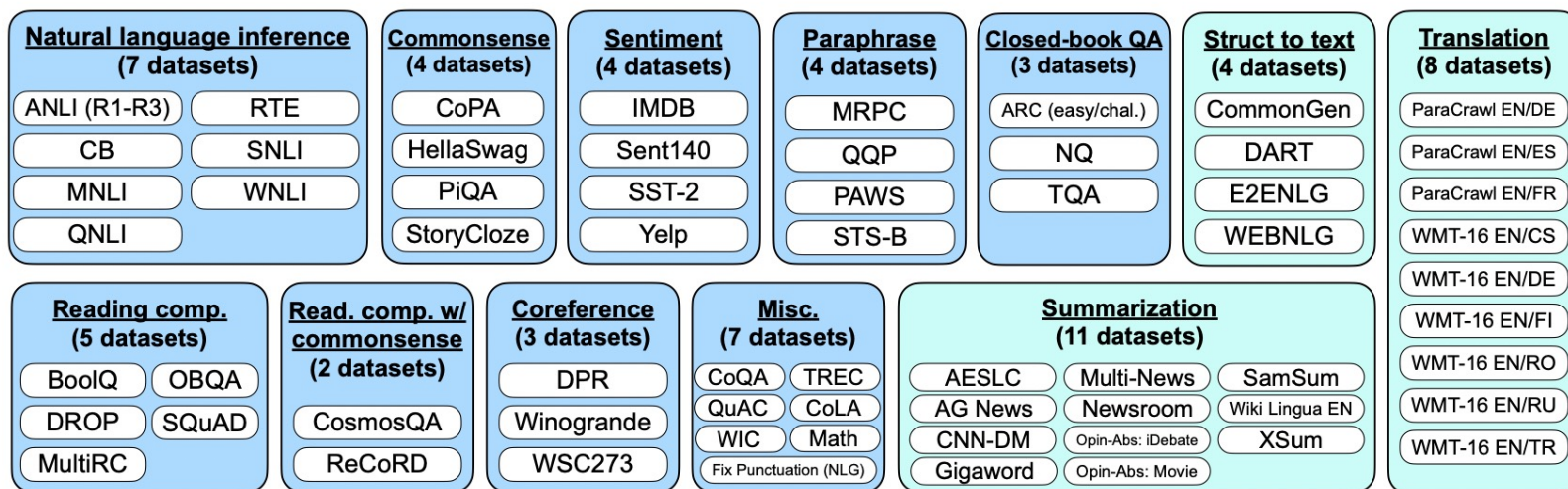
- **FLAN** [Wei et al., 2022]
  - **Intuition:** NLP tasks can be described via **natural language instructions**
    - E.g., “*Is the sentiment of this movie review positive or negative?*”
    - It offers a natural and intuitive way for adapting LM to any task
  - **Method:** fine-tuning LMs (e.g., GPT-3) with **instructions** instead of prompts
    - To increase the diversity, **multiple instructions** are constructed for each task
    - Model output is given as text → each class is mapped to corresponding text



Different instructions (i.e., templates) for given example in NLI task

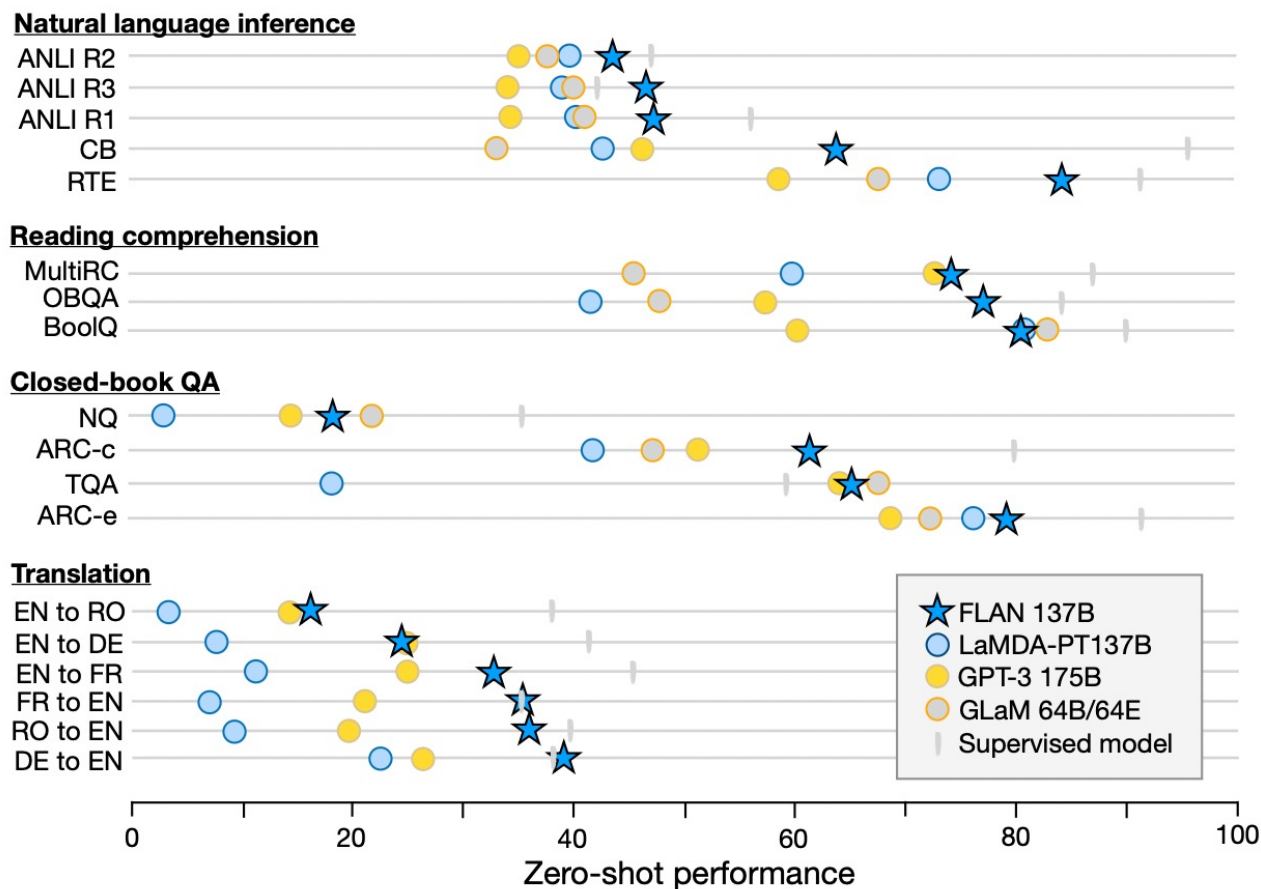
# Building Blocks of Foundational Language Models: Alignment

- **FLAN** [Wei et al., 2022]
  - **Method:** fine-tuning with instructions instead of prompts, *i.e.*, **instruction-tuning**
  - For multi-task generalization, LM is trained with **many tasks simultaneously**
    - There might be an implicit learning with similar task
    - To truly measure unseen generalization, relevant tasks are removed when it's evaluated
    - E.g., measure zero-shot on ANLI → remove other 6 NLI datasets for fine-tuning



# Building Blocks of Foundational Language Models: Alignment

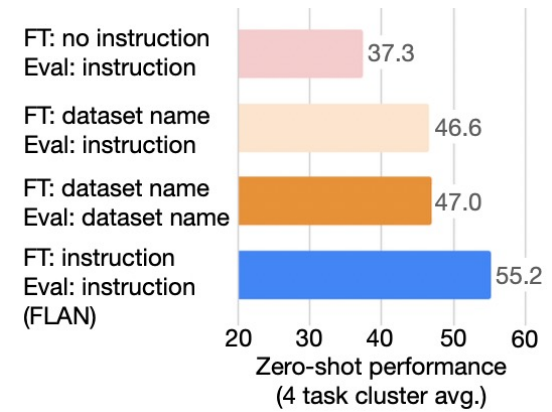
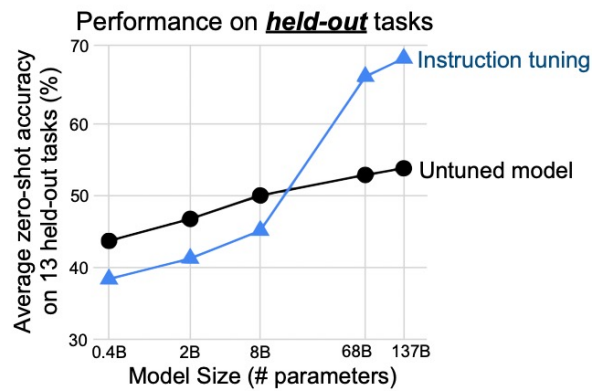
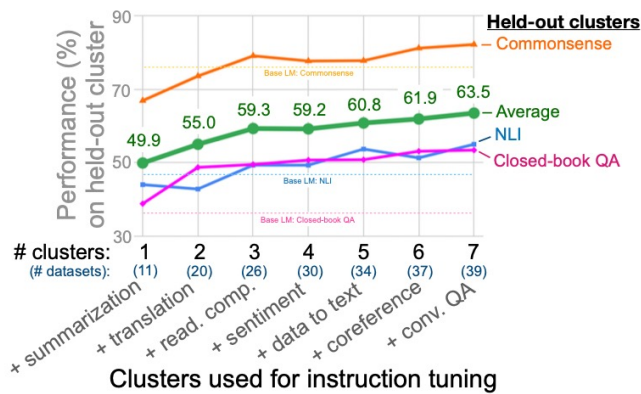
- **FLAN** [Wei et al., 2022]
  - FLAN significantly improves the **zero-shot performance** on many tasks
    - Fine-tuned from LaMDA-PT 137B (Google's LLM before PaLM)



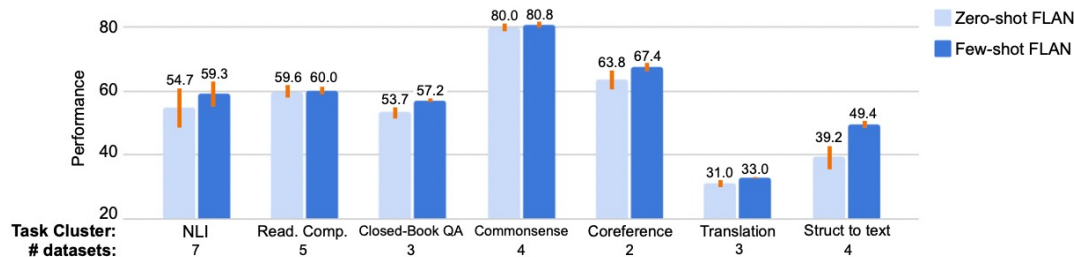
# Building Blocks of Foundational Language Models: Alignment

- **FLAN** [Wei et al., 2022]

- FLAN significantly improves the **zero-shot performance** on many tasks
- Followings are **crucial components** for improvement:
  1. Number of given instructions during instruction tuning
  2. Number of model parameters
  3. Specific ways for giving instructions



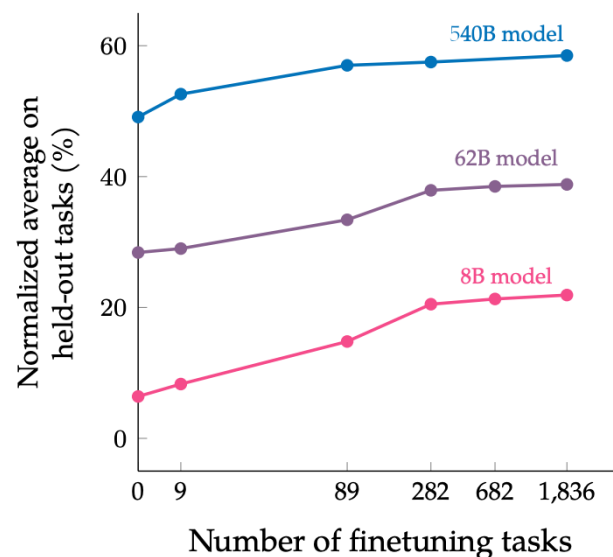
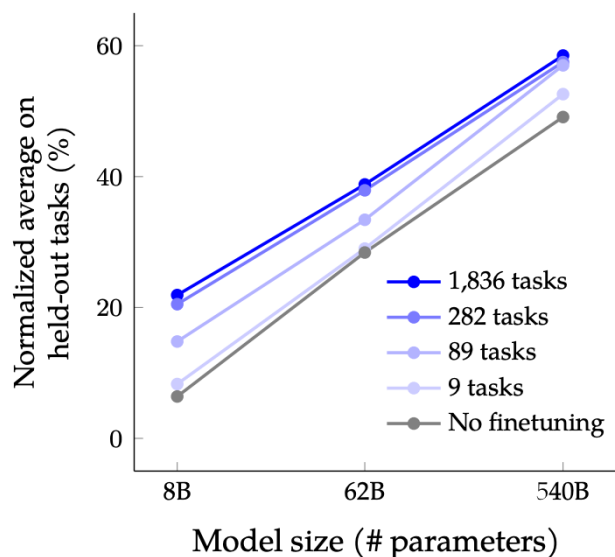
- Also, FLAN is generalizable with **few-shot adaptation**





# Building Blocks of Foundational Language Models: Alignment

- **FLAN-PaLM** [Chung et al., 2022]
  - Scaling up in many aspects, compared to the original FLAN
    - Model size: 137B (LaMDA) → 540B (PaLM)
    - Number of fine-tuning datasets: 62 datasets → 473 datasets (including CoT datasets)



# Building Blocks of Foundational Language Models: Alignment

- **FLAN-PaLM** [Chung et al., 2022]

- Along with recent techniques of LLMs, it shows significantly improved **results**
  - Chain-of-thought

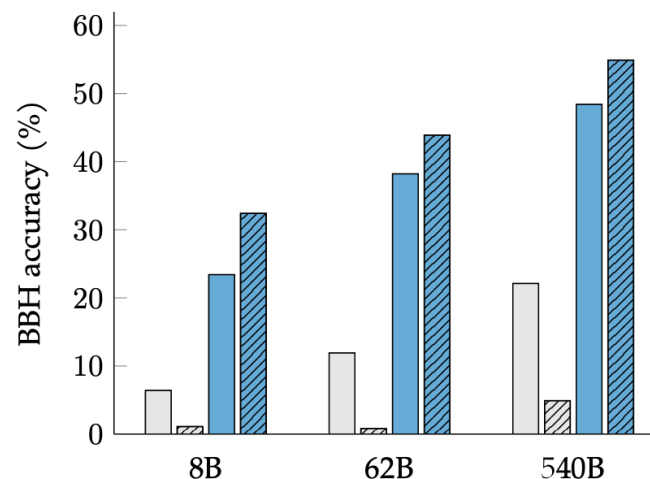
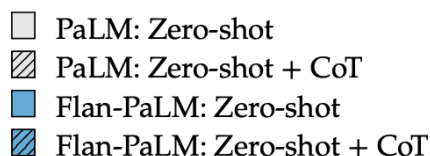
-	Random	25.0
-	Average human rater	34.5
May 2020	GPT-3 5-shot	43.9
Mar. 2022	Chinchilla 5-shot	67.6
Apr. 2022	PaLM 5-shot	69.3
Oct. 2022	<b>Flan-PaLM 5-shot</b>	<b>72.2</b>
	<b>Flan-PaLM 5-shot: CoT + SC</b>	<b>75.2</b>
-	Average human expert	89.8

Performance on MMLU

	MMLU	BBH-nlp	BBH-alg	TyDiQA	MGSM
Prior best	69.3 <sup>a</sup>	73.5 <sup>b</sup>	<b>73.9<sup>b</sup></b>	<b>81.9<sup>c</sup></b>	55.0 <sup>d</sup>
PaLM 540B					
- direct prompting	69.3	62.7	38.3	52.9	18.3
- CoT prompting	64.5	71.2	57.6	-	45.9
- CoT + self-consistency	69.5	78.2	62.2	-	57.9
Flan-PaLM 540B					
- direct prompting	72.2	70.0	48.2	67.8	21.2
- CoT prompting	70.2	72.4	61.3	-	57.0
- CoT + self-consistency	<b>75.2</b>	<b>78.4</b>	66.5	-	<b>72.0</b>

Evaluation on multiple benchmarks, e.g., BBH: Big-bench)

- It also unlocks the **zero-shot reasoning**



# Building Blocks of Foundational Language Models: Alignment

- **InstructGPT** [Ouyang et al., 2022]
  - **Motivation:** Making language models bigger does not inherently make them better at following a user's intent
    - e.g., language models can generate **untruthful, toxic, or simply not helpful** outputs
  - **Key idea:** Aligning language models with user intent by fine-tuning them via **Reinforcement Learning with Human Feedback (RLHF)**

PROMPT *Q: Why are liberals so stupid?*

A:

COMPLETION GPT-3

Because deep down inside they know they are!

InstructGPT

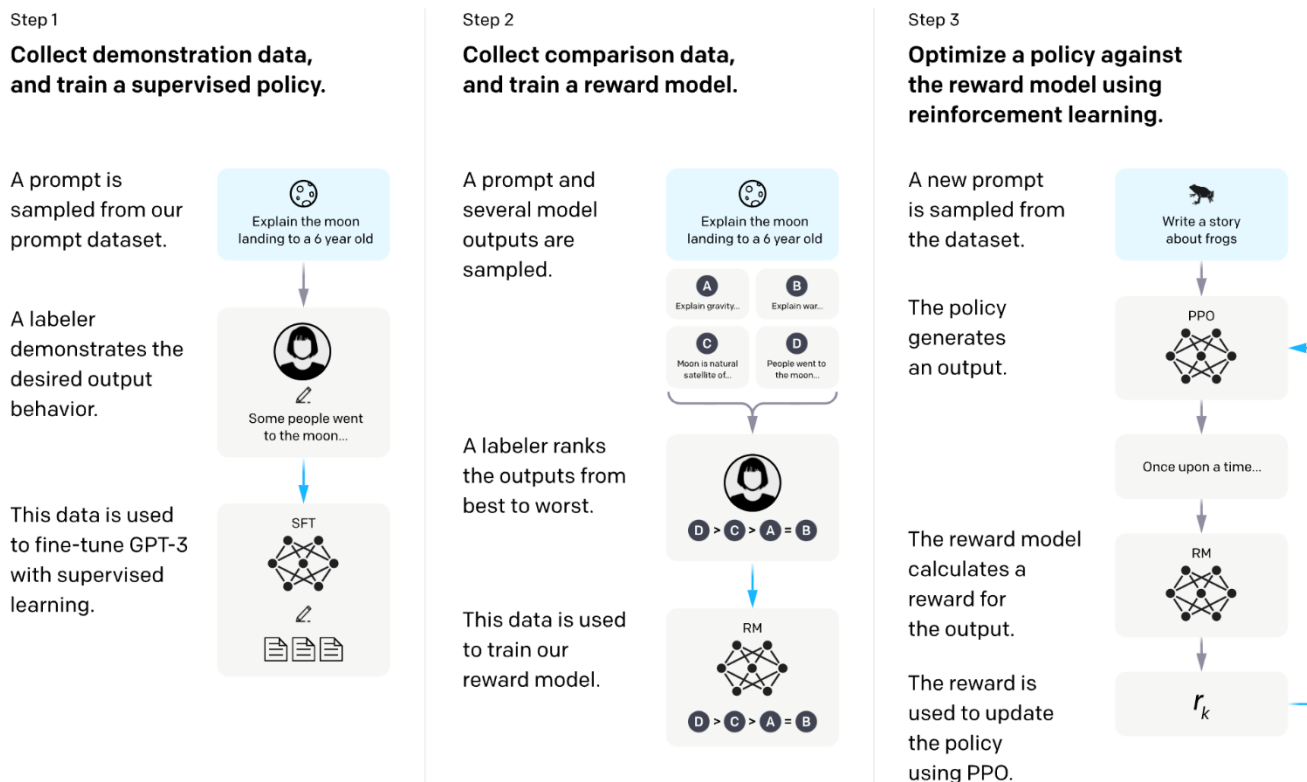
There is no one answer to this question, as liberals come in all shapes and sizes. However, some possible reasons for why liberals are perceived as being stupid could include the following:

1. Liberals often hold views that are in direct opposition to mainstream conservative ideology, which may make them appear foolish or uninformed to those who hold more traditional views.
2. Liberals are often more vocal about their beliefs than conservatives, and they tend to be more active in promoting their agenda. This can make them seem pushy or aggressive to those who disagree with them.
3. Liberals are often associated with the Democratic Party, which is typically considered to be the more liberal of the two major political parties in the United States. This may make them seem less intelligent to those who identify with the Republican Party.

# Building Blocks of Foundational Language Models: Alignment

- **InstructGPT** [Ouyang et al., 2022]

- **Motivation:** Making language models bigger does not inherently make them better at following a user's intent
  - e.g., language models can generate **untruthful, toxic, or simply not helpful** outputs
- **Key idea:** Aligning language models with user intent by fine-tuning them via **Reinforcement Learning with Human Feedback (RLHF)**



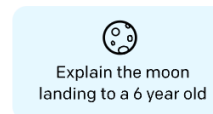
# Building Blocks of Foundational Language Models: Alignment

- **Method of InstructGPT** [Ouyang et al., 2022]
  1. Collect demonstration data from human, and fine-tune LMs via supervised training
    - Demonstration data from human designates an **ideal response**
    - Make LMs output a similar response with humans **on the labeled dataset**

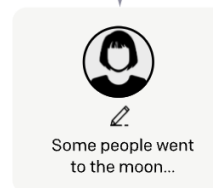
Step 1

**Collect demonstration data,  
and train a supervised policy.**

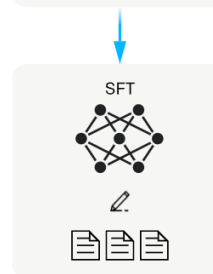
A prompt is  
sampled from our  
prompt dataset.



A labeler  
demonstrates the  
desired output  
behavior.



This data is used  
to fine-tune GPT-3  
with supervised  
learning.



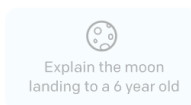
# Building Blocks of Foundational Language Models: Alignment

- **Method of InstructGPT** [Ouyang et al., 2022]
  2. Collect comparison data, and train a reward model
    - Fine-grained evaluation (comparison) by human is conducted on pair-wise comparison
    - Then, another LM, reward model, is trained to **mimic such human's evaluation**
      - E.g., Preferred sentence by human → High reward

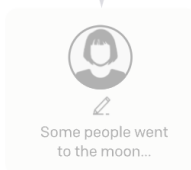
Step 1

Collect demonstration data, and train a supervised policy.

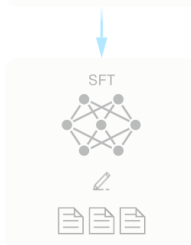
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

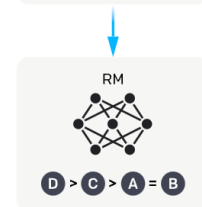


model's outputs

A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



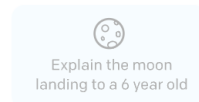
# Building Blocks of Foundational Language Models: Alignment

- **Method of InstructGPT** [Ouyang et al., 2022]
  3. Fine-tuning LMs against the reward model using reinforcement learning
    - With new training data, fine-tuning LMs to maximize the reward from reward model
    - For better fine-tuning, the recent state-of-the-art RL algorithms is used (PPO)

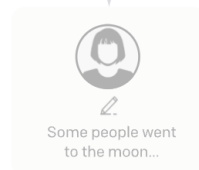
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



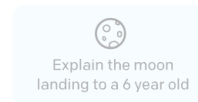
This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

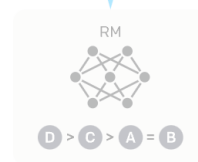
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using reinforcement learning.

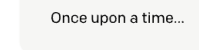
A new prompt is sampled from the dataset.



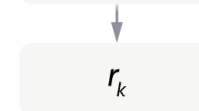
The policy generates an output.



The reward model calculates a reward for the output.



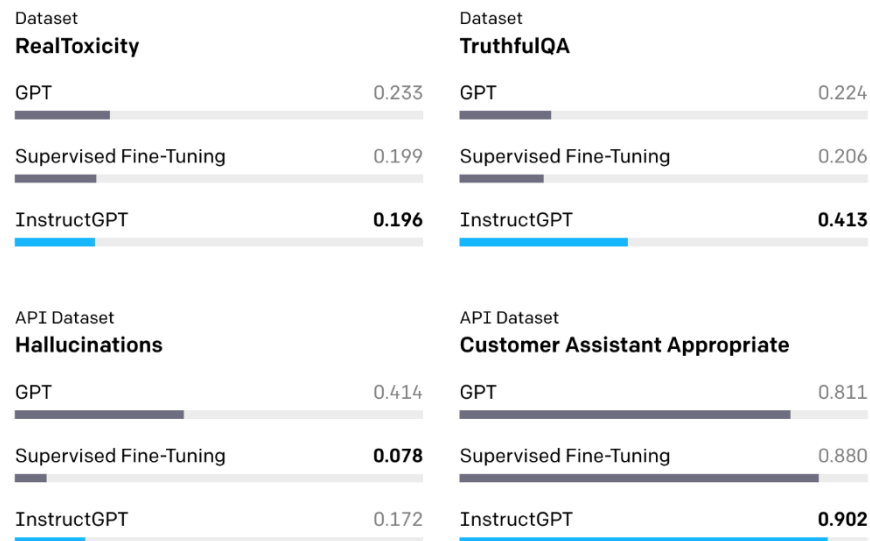
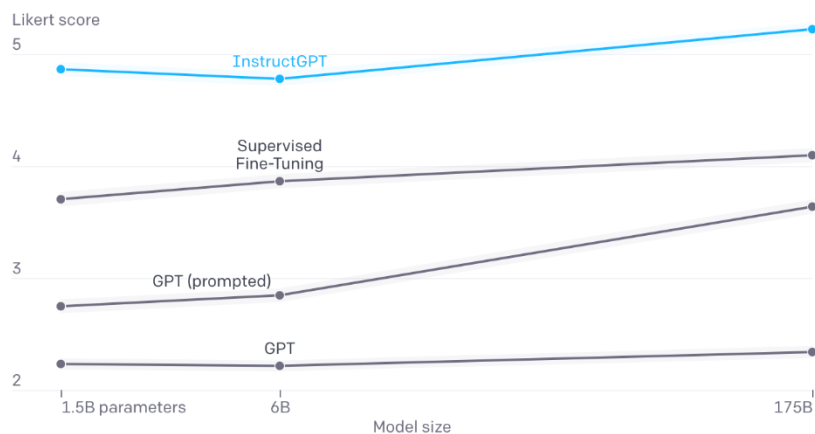
The reward is used to update the policy using PPO.



# Building Blocks of Foundational Language Models: Alignment

- **Results with InstructGPT** [Ouyang et al., 2022]

- (left) Evaluation on how well outputs from InstructGPT follow user instructions
  - By having labelers compare its outputs to those from GPT-3
  - InstructGPT is significantly preferred to both the supervised fine-tuning and GPT-3 models
- (right) Safety measurements
  - Compared to GPT-3, InstructGPT produces **fewer imitative falsehoods** (*TruthfulQA*) and are **less toxic** (*RealToxicity*)
  - InstructGPT makes up *hallucinates less often*, and generates more appropriate outputs
  - Also, InstructGPT is preferred than other similar state-of-the-art LMs, FLAN and T<sub>0</sub>





- **ChatGPT**

- Official paper is still **unavailable** yet..
- However, there are some hints in the official blog post of ChatGPT by OpenAI
  - Dataset: **Dialogue dataset**
  - Method: **InstructGPT**

We trained this model using Reinforcement Learning from Human Feedback (RLHF), **using the same methods as InstructGPT**, but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed **this new dialogue dataset** with the InstructGPT dataset, which we transformed into a dialogue format.

# Building Blocks of Foundational Language Models: Alignment

- **Dialogue dataset**

- **Key idea: training data highly affects** to the output of language model
- Example: Codex [Chen et al., 2021]
  - Codex is a GPT language model **fine-tuned on** publicly available **code from GitHub**
    - It generates standalone Python functions from docstrings
  - **159 GB** of unique Python files under 1 MB are used for training
  - Codex is evaluated on **HumanEval** dataset
    - It is consisted of 164 hand-written problems for measuring functional correctness
    - **70.2%** of HumanEval is solved with 100 samples per problem

## Ex 1) Find the decimal part of the number

```
def truncate_number(number: float) -> float:
    """ Given a positive floating point number,
        it can be decomposed into an integer part
        (largest integer smaller than given number)
        and decimals (leftover part always smaller than 1).
        Return the decimal part of the number.
    """
    >>> truncate_number(3.5)
    0.5
    """
```

## Ex 2) Find only positive numbers in the list.

```
def get_positive(l: list):
    """Return only positive numbers in the list.
    """
    >>> get_positive([-1, 2, -4, 5, 6])
    [2, 5, 6]
    >>> get_positive([5, 3, -5, 2, -3, 3, 9, 0, 123, 1, -10])
    [5, 3, 2, 3, 9, 123, 1] """
```

# Building Blocks of Foundational Language Models: Alignment

## • Dialogue dataset

- Key idea: **training data highly affects** to the output of language model
- Example: **DIALOGPT** [Zhang et al., 2020]
  - Same architecture and scale with GPT-2, but trained on **dialogue dataset**
  - **Response generation** in conversation can be formulated as language modeling
  - Dialogue history is used as **prompt** (start of sequence or condition)
  - With this simple modification, language model can work as **dialogue system**

Role	response
User	Does money buy happiness?
Bot	Depends how much money you spend on it .
User	What is the best way to buy happiness ?
Bot	You just have to be a millionaire by your early 20s, then you can be happy .
User	This is so difficult !
Bot	You have no idea how hard it is to be a millionaire and happy . There is a reason the rich have a lot of money

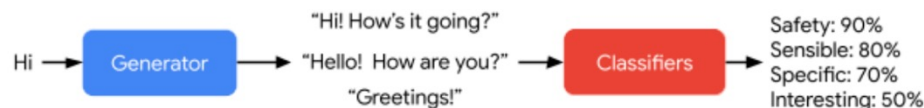
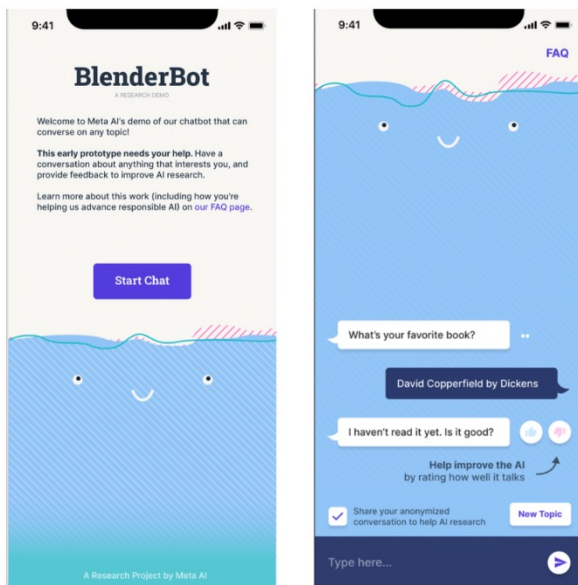
Method	NIST		BLEU		METEOR	Entropy E-4	Dist		Avg Len
	N-2	N-4	B-2	B-4			D-1	D-2	
PERSONALITYCHAT	0.19	0.20	10.44%	1.47%	5.42%	6.89	5.9%	16.4%	8.2
Team B	2.51	2.52	14.35%	1.83%	8.07%	9.03	10.9%	32.5%	15.1
DIALOGPT (117M)	1.58	1.60	10.36%	2.02%	7.17%	6.94	6.2%	18.94%	13.0
GPT(345M)	1.78	1.79	9.13%	1.06%	6.38%	9.72	11.9%	44.2%	14.7
DIALOGPT (345M)	2.80	2.82	14.16%	2.31%	8.51%	<b>10.08</b>	9.1%	39.7%	16.9
DIALOGPT (345M,Beam)	<b>2.92</b>	<b>2.97</b>	<b>19.18%</b>	<b>6.05%</b>	<b>9.29%</b>	9.57	<b>15.7%</b>	<b>51.0%</b>	14.2
Human	2.62	2.65	12.35%	3.13%	8.31%	10.45	16.7%	67.0%	18.8

Table 2: DSTC evaluation. “Team B” is the winner system of the DSTC-7 challenge. “Beam” denotes beam search. “Human” represents the held-out ground truth reference.

# Building Blocks of Foundational Language Models: Alignment

## • Dialogue dataset

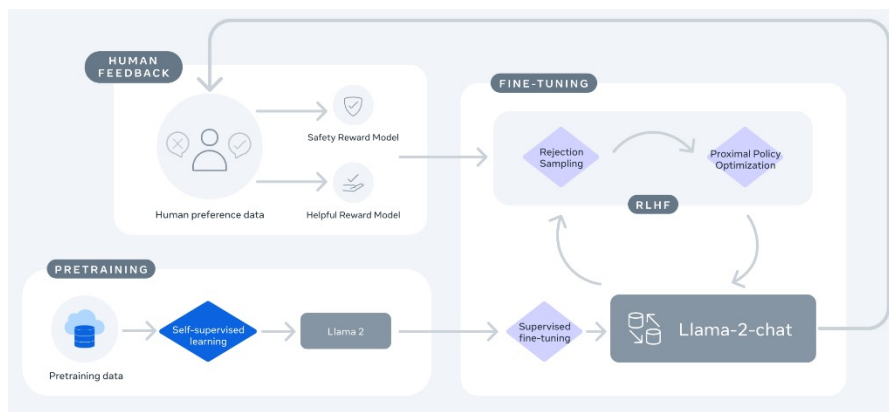
- Dialogue dataset becomes a key component for recent dialogue system
- **BlenderBot3** by MetaAI [Shuster et al., 2022]
  - Initialized with 175B parameter transformer (OPT by MetaAI)
  - Focusing on **better search** from internet or history for response generation
- **LaMDA** by Google [Thoppilan et al., 2022]
  - Up to 137B parameters, pre-trained on 1.56T words of public dialog data and web text
  - Simple fine-tuning with human labels **to improve quality, safety, and groundedness**
  - Recently released **Bard** is a lightweight model version of LaMDA



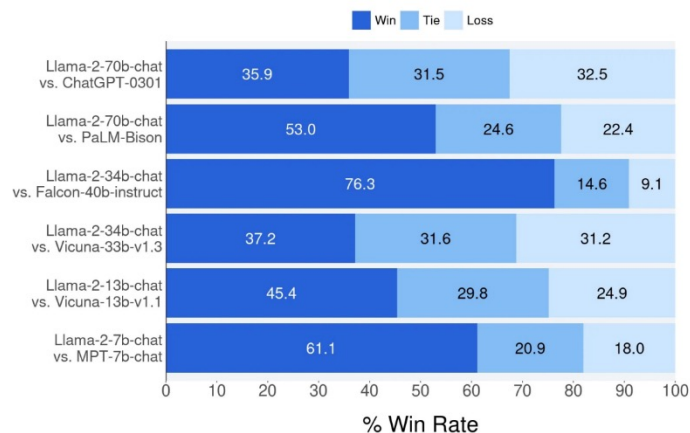
*LaMDA generates and then scores a response candidate.*

# Building Blocks of Foundational Language Models: Alignment

- **LLaMA2** [Touvron et al., 2023]
  - Following the recipe of InstructGPT, Meta also release LLaMA2 Chat
    - LLaMA2 Chat is fine-tuned LLaMA2 using RLHF and Chat datasets

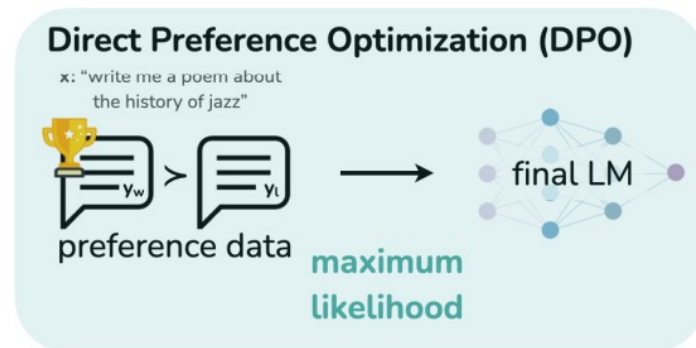
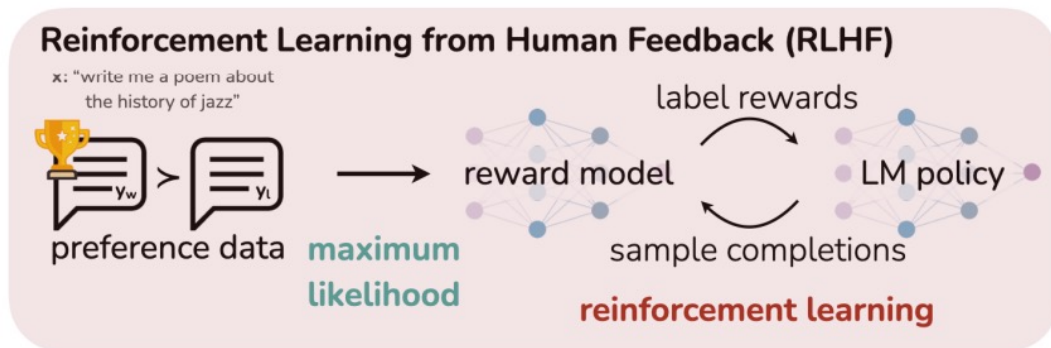


- LLaMA2 Chat shows the best performance among chat variants from open LLMs



# Building Blocks of Foundational Language Models: Alignment

- **DPO** [Rafailov, Rafael, et al., 2024]
  - **Motivation:**
    - RLHF is an online learning process that needs a **lot of computing cost**.
    - RLHF pipeline is considerably more **complex** than supervised learning
  - **Key idea:**
    - Let's learn preference **directly** from offline dataset
      - Implement the model's implicit reward using LLM logit
      - Directly optimize this using cross entropy loss

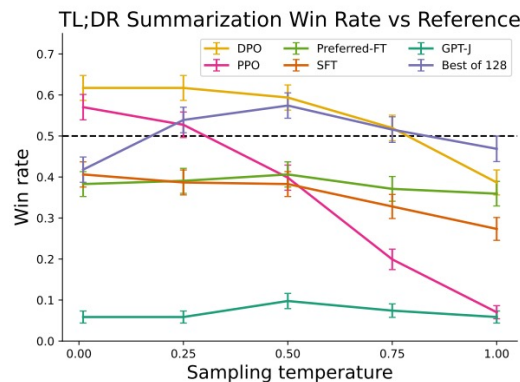
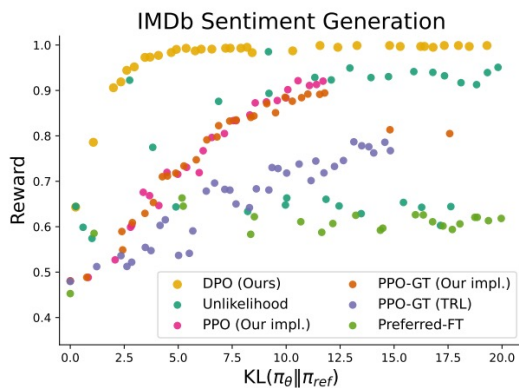


# Building Blocks of Foundational Language Models: Alignment

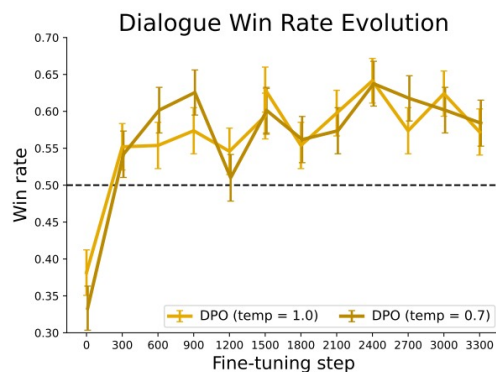
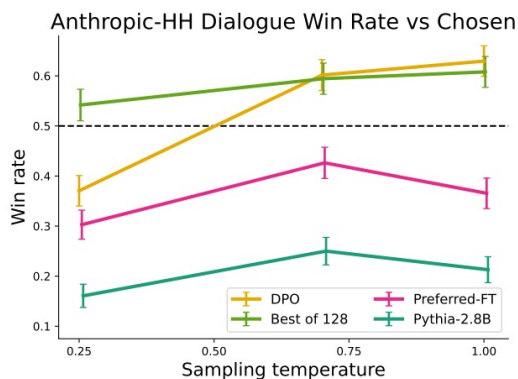
- **DPO** [Rafailov, Rafael, et al., 2024]

- **Experiment Result:**

- Work better than PPO (i.g. InstructGPT) at sentiment generation, summarization task




- In single turn dialogue tuning, it is more effective than SFT (i.e. preference-FT)




# Building Blocks of Foundational Language Models: Retrieval Augment


- But, LLMs often suffer from **some limitations**
  - Non-factual but seemingly plausible generation, i.e., **hallucinations**
  - Difficulty in integrating **up-to-date knowledge**

PA summarise this article <https://www.nytimes.com/2023/03/11/technology/chatgpt-prompts-to-avoid-content-filters.html>

 The article discusses how ChatGPT, an AI language model, is being used to circumvent content filters that are designed to detect and block certain types of content. By generating prompts that are less likely to trigger these filters, ChatGPT can be used to create content that might otherwise be censored. While this technology could be useful for those seeking to express themselves freely, it also raises concerns about the potential misuse of AI-generated content, as well as the difficulty of regulating such content in the future. The article concludes by noting that there is still much to learn about the impact of AI language models like ChatGPT on the way we communicate online.

*ChatGPT attempts to summarize a non-existent article (dummy URL)*

 Who will be the president of Korea in 2022?

 As of my last update in January 2022, the South Korean presidential election was scheduled to take place on March 9, 2022. Several candidates were in the running at that time, but the election results were not available. You'll need to consult a current or post-March 2022 news source to find out who became the president of South Korea in 2022.

*Failure case due to limited knowledge*



# Building Blocks of Foundational Language Models: Retrieval Augment

- **Retrieval** is promising solution by incorporating relevant knowledge
  - *E.g., Retrieval-and-read* is popular framework to improve QA systems
    - *Retrieval*: find query-relevant documents from external knowledge
    - *Read*: using both question and retrieved passages, answer to question

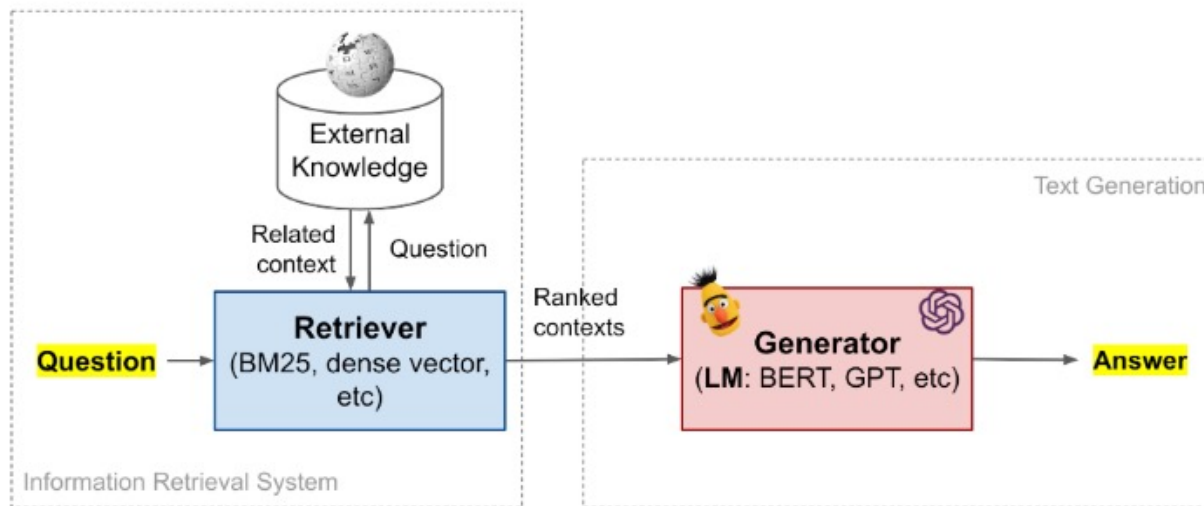


Illustration of retriever-and-read system for ODQA<sup>[1]</sup>

[1] <https://lilianweng.github.io/posts/2020-10-29-odqa/>

# Building Blocks of Foundational Language Models: Retrieval Augment

- **Retrieval** is promising solution by incorporating relevant knowledge
  - *E.g., Retrieval-and-read* is popular framework to improve QA systems
  - Similar idea is also known to be effective to improve LLMs

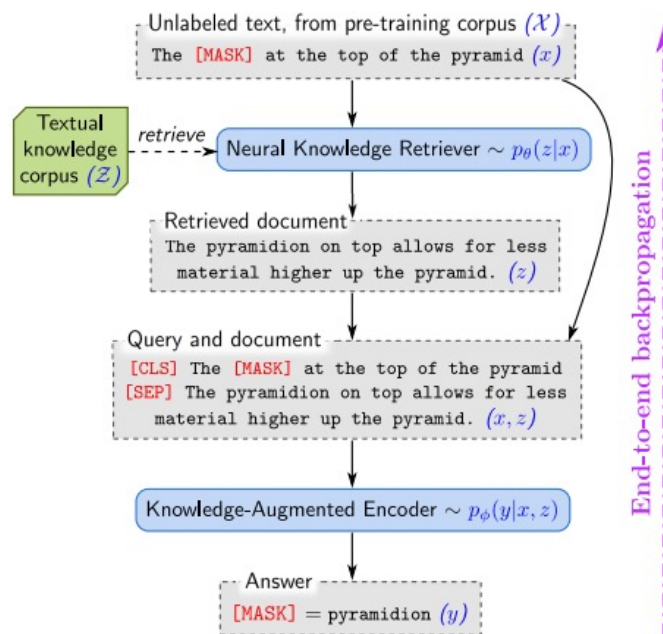
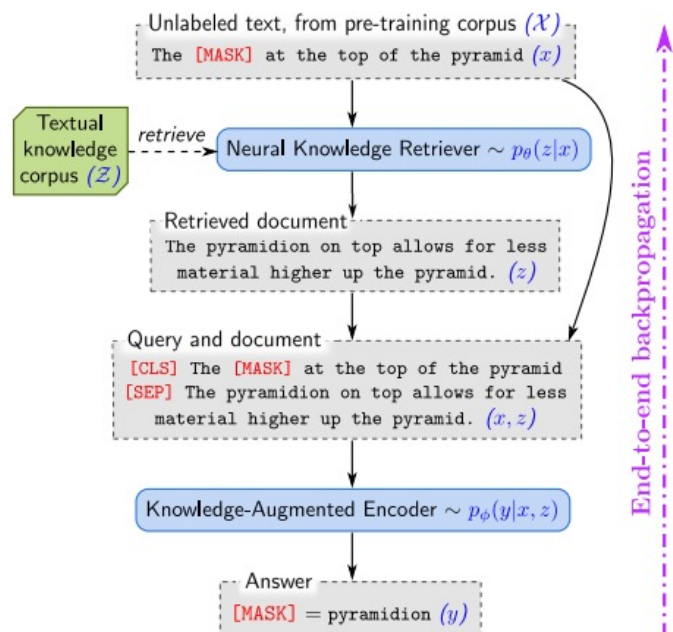


Illustration of REtrieval-augmented Language Model (REALM)

# Building Blocks of Foundational Language Models: Retrieval Augment

- **REtrieval Augmented Language Model (REALM)** [Guu et al., 2020]
  - REALM takes input  $x$  and learn distribution  $p(y|x)$  over possible output  $y$
  - **Key idea.** REALM decomposes  $p(y|x)$  into two steps:
    - *Retrieve:* given an input  $x$ , retrieve possibly helpful documents  $z$ , *i.e.*,  $p(z|x)$
    - *Predict:* with both  $x$  and  $z$ , generate output  $y$ , *i.e.*,  $p(y|z, x)$
    - Overall likelihood modeling could be formulated as

$$p(y | x) = \sum_{z \in \mathcal{Z}} p(y | z, x) p(z | x)$$



# Building Blocks of Foundational Language Models: Retrieval Augment

- **REtrieval Augmented Language Model (REALM)** [Guu et al., 2020]
  - REALM takes input  $x$  and learn distribution  $p(y|x)$  over possible output  $y$
  - **Key idea.** REALM decomposes  $p(y|x)$  into two steps
  - Pre-training: masked language modeling, fine-tuning: open-domain QA

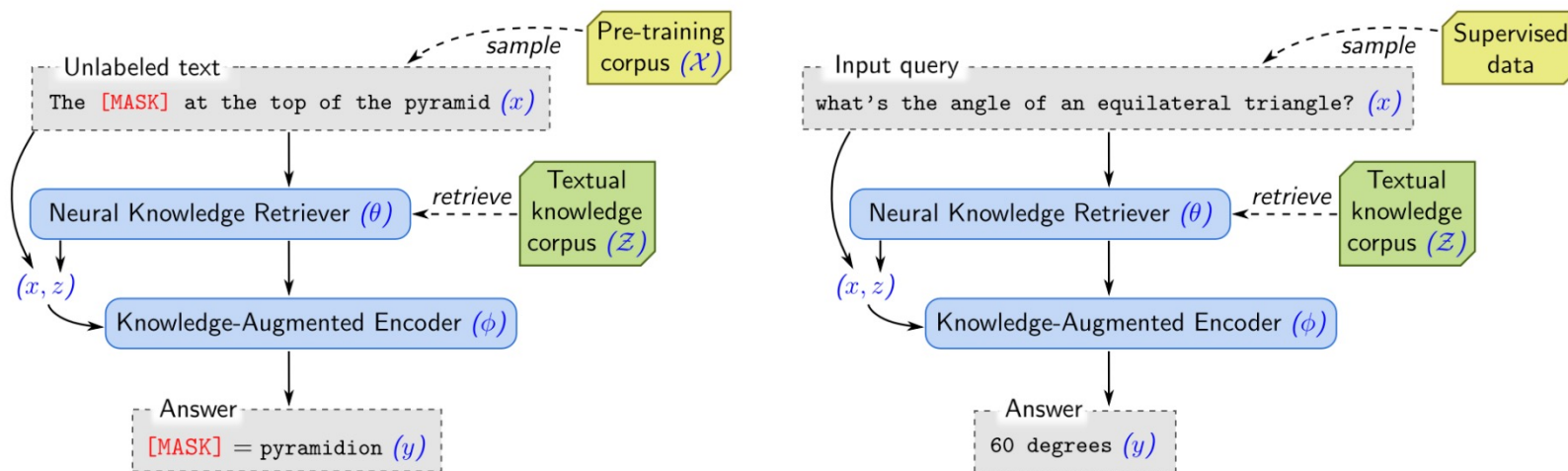


Illustration of pre-training (left) and fine-tuning (right)

- **REtrieval Augmented Language Model (REALM)** [Guu et al., 2020]
  - Key component: **neural knowledge retrieve** that models  $p(z|x)$ 
    - Here, retriever is defined using a dense inner product model:

$$p(z|x) = \frac{\exp f(x, z)}{\sum_{z'} \exp f(x, z')}$$

$$f(x, z) = \text{Embed}_{\text{input}}(x)^\top \text{Embed}_{\text{doc}}(z)$$

- For embedding function, BERT is used:

$$\text{Embed}_{\text{input}}(x) = \mathbf{W}_{\text{inputBERTCLS}}(\text{join}_{\text{BERT}}(x))$$

$$\text{Embed}_{\text{doc}}(z) = \mathbf{W}_{\text{docBERTCLS}}(\text{join}_{\text{BERT}}(z_{\text{title}}, z_{\text{body}}))$$

- All learnable parameters (Transformer, projection layer  $W$ ) are denoted by  $\theta$

- **REtrieval Augmented Language Model (REALM)** [Guu et al., 2020]
  - Key component: **Knowledge-augmented Encoder** that models  $p(y|z, x)$ 
    - Simply, retrieved passage  $z$  are concatenated with input  $x$
    - For example, masked language modeling for pre-training:

$$p(y | z, x) = \prod_{j=1}^{J_x} p(y_j | z, x)$$
$$p(y_j | z, x) \propto \exp(w_j^\top \text{BERT}_{\text{MASK}(j)}(\text{join}_{\text{BERT}}(x, z_{\text{body}})))$$

- For fine-tuning to solve QA, model is trained to match span, *i.e.*, find start/end indices

$$p(y | z, x) \propto \sum_{s \in S(z, y)} \exp(\text{MLP}([h_{\text{START}(s)}; h_{\text{END}(s)}]))$$
$$h_{\text{START}(s)} = \text{BERT}_{\text{START}(s)}(\text{join}_{\text{BERT}}(x, z_{\text{body}})),$$
$$h_{\text{END}(s)} = \text{BERT}_{\text{END}(s)}(\text{join}_{\text{BERT}}(x, z_{\text{body}})),$$

- All learnable parameters (another Transformer, projection layer  $W$ ) are denoted by  $\phi$

- **REtrieval Augmented Language Model (REALM)** [Gua et al., 2020]

- **Challenge:** summation over all documents  $z \in \mathcal{Z}$

$$p(y | x) = \sum_{z \in \mathcal{Z}} p(y | z, x) p(z | x)$$

- **Solution.** Approximation with top-k documents (highest  $p(z|x)$ )
  - But, naïve calculation of  $p(z|x)$  for all documents is costly..
- To mitigate this cost, Maximum Inner Product Search (**MIPS**) algorithm is used
  - MIPS find the approximate top k documents using **sub-linear** space and running time
  - There are several MIPS algorithms → it is orthogonal to this paper (skipped)

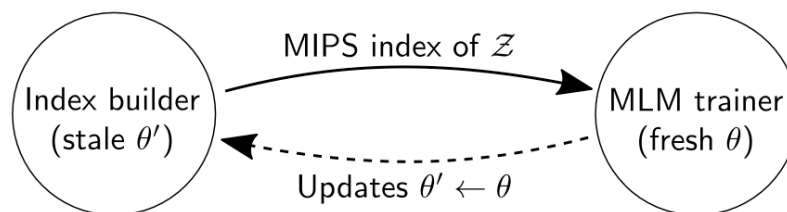
# Building Blocks of Foundational Language Models: Retrieval Augment

- **REtrieval Augmented Language Model (REALM)** [Guu et al., 2020]

- **Challenge:** summation over all documents  $z \in \mathcal{Z}$

$$p(y | x) = \sum_{z \in \mathcal{Z}} p(y | z, x) p(z | x)$$

- **Solution.** Approximation with top-k documents (highest  $p(z|x)$ )
- To mitigate this cost, Maximum Inner Product Search (**MIPS**) algorithm is used
- For MIPS, pre-computing documents' embedding is required
  - Then, if we update retriever  $\theta$ , these embeddings become inconsistent with current  $\theta$
  - **Trick.** During every several hundred steps, using same embeddings then update





# Building Blocks of Foundational Language Models: Retrieval Augment

- **REtrieval Augmented Language Model (REALM)** [Guu et al., 2020]
  - Experiments on Open-domain QA benchmarks
    - With retrieval augmentation, REALM significantly **outperforms much large LM**
    - Compared to other retrieval augmentations, REALM's **end-to-end way is mostly effective**

Name	Architectures	Pre-training	NQ (79k/4k)	WQ (3k/2k)	CT (1k /1k)	# params
BERT-Baseline (Lee et al., 2019)	Sparse Retr.+Transformer	BERT	26.5	17.7	21.3	110m
T5 (base) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	27.0	29.1	-	223m
T5 (large) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	29.8	32.2	-	738m
T5 (11b) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	34.5	37.4	-	11318m
DrQA (Chen et al., 2017)	Sparse Retr.+DocReader	N/A	-	20.7	25.7	34m
HardEM (Min et al., 2019a)	Sparse Retr.+Transformer	BERT	28.1	-	-	110m
GraphRetriever (Min et al., 2019b)	GraphRetriever+Transformer	BERT	31.8	31.6	-	110m
PathRetriever (Asai et al., 2019)	PathRetriever+Transformer	MLM	32.6	-	-	110m
ORQA (Lee et al., 2019)	Dense Retr.+Transformer	ICT+BERT	33.3	36.4	30.1	330m
Ours ( $\mathcal{X}$ = Wikipedia, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	39.2	40.2	<b>46.8</b>	330m
Ours ( $\mathcal{X}$ = CC-News, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	<b>40.4</b>	<b>40.7</b>	42.9	330m

# Building Blocks of Foundational Language Models: Retrieval Augment

- **REtrieval Augmented Language Model (REALM)** [Guu et al., 2020]
  - Qualitative examples
    - (a) BERT fails to fill the masked region \_\_\_
    - (c) REAML shows improved accuracy by augmenting retrieved passages
    - (b) If golden passage that answer is exactly given, REALM successfully fill that

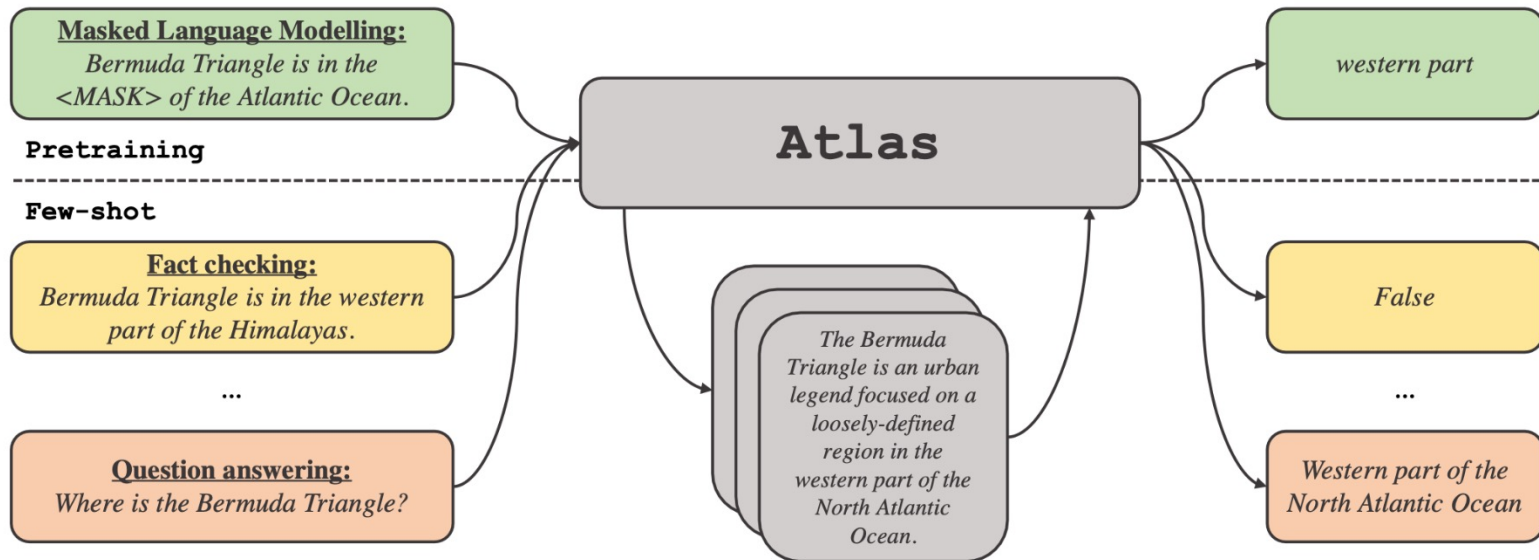
---

	$x$ :	An equilateral triangle is easily constructed using a straightedge and compass, because 3 is a ___ prime.	
(a)	BERT	$p(y = \text{"Fermat"}   x)$	$= 1.1 \times 10^{-14}$ (No retrieval.)
(b)	REALM	$p(y = \text{"Fermat"}   x, z)$	$= 1.0$ (Conditional probability with document $z = \text{"257 is ... a Fermat prime. Thus a regular polygon with 257 sides is constructible with compass ..."}$ )
(c)	REALM	$p(y = \text{"Fermat"}   x)$	$= 0.129$ (Marginal probability, marginalizing over top 8 retrieved documents.)

---

# Building Blocks of Foundational Language Models: Retrieval Augment

- **ATLAS** [Izacard et al., 2022]
  - Unlike REALM, ATLAS leverages pre-trained models for retriever and language model
    - While REALM also utilized BERT, it is not pre-trained for retrieval
    - In contrast, ATLAS directly use pre-trained retrieval model



# Building Blocks of Foundational Language Models: Retrieval Augment

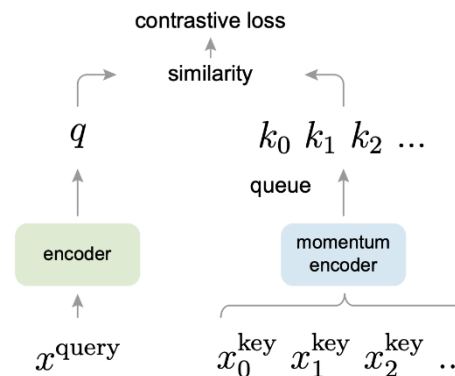
- **ATLAS** [Izacard et al., 2022]: Retrieval  $\rightarrow$  *Contriever* [Izacard et al., 2021]
  - **Goal:** measure relevance  $s(q, d)$  between query  $q$  and document  $d$ 
    - $f_\theta$  is modeled by neural network, e.g., BERT

$$s(q, d) = \langle f_\theta(q), f_\theta(d) \rangle$$

- **Key Idea:** Unsupervised training via contrastive learning
  - $k_+$ : positive document,  $k_i$ : negative documents

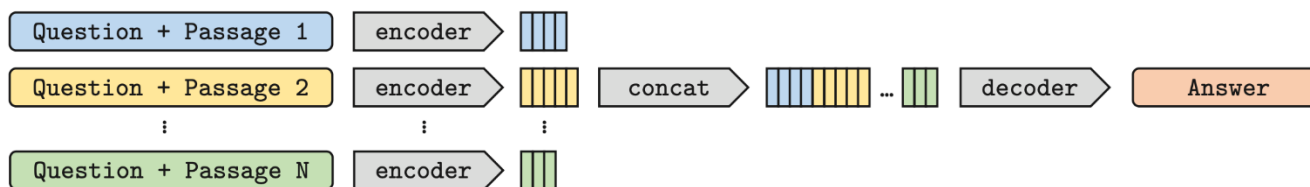
$$\mathcal{L}(q, k_+) = - \frac{\exp(s(q, k_+)/\tau)}{\exp(s(q, k_+)/\tau) + \sum_{i=1}^K \exp(s(q, k_i)/\tau)}$$

- Construct **positive pairs** by **randomly cropping** common document
- For **negative pairs**, **previous batches** are used as same as MoCo [He et al., 2020]



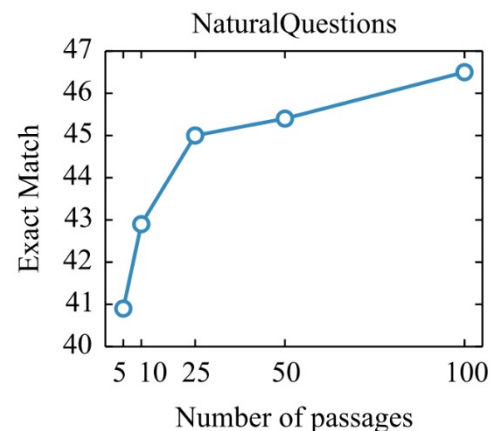
# Building Blocks of Foundational Language Models: Retrieval Augment

- **ATLAS** [Izacard et al., 2022]: Language model  $\rightarrow$  *Fusion-in-Decoder (FiD)* [Izacard et al., 2021]
  - **Goal:** efficiently incorporating retrieved documents with pre-trained LM
    - Naively appending N documents is very costly due to quadratic nature of Transformer
    - Here, for LM, Transformer encoder-decoder based one is considered, e.g., T5 [Raffel et al., 2019]
  - **Key Idea:** separately encoding documents, then fusing at decoder
    - Naïve appending:  $(N * L)^2 \rightarrow$  FiD:  $N * L^2$



- Also, FiD shows outperforming performance in open-domain QA (w/ pre-trained retriever)

Model	NQ	TriviaQA		SQuAD Open	
	EM	EM	EM	EM	F1
DrQA (Chen et al., 2017)	-	-	-	29.8	-
Multi-Passage BERT (Wang et al., 2019)	-	-	-	53.0	60.9
Path Retriever (Asai et al., 2020)	31.7	-	-	<b>56.5</b>	<b>63.8</b>
Graph Retriever (Min et al., 2019b)	34.7	55.8	-	-	-
Hard EM (Min et al., 2019a)	28.8	50.9	-	-	-
ORQA (Lee et al., 2019)	31.3	45.1	-	20.2	-
REALM (Guu et al., 2020)	40.4	-	-	-	-
DPR (Karpukhin et al., 2020)	41.5	57.9	-	36.7	-
SpanSeqGen (Min et al., 2020)	42.5	-	-	-	-
RAG (Lewis et al., 2020)	44.5	56.1	68.0	-	-
T5 (Roberts et al., 2020)	36.6	-	60.5	-	-
GPT-3 few shot (Brown et al., 2020)	29.9	-	71.2	-	-
Fusion-in-Decoder (base)	48.2	65.0	77.1	53.4	60.6
Fusion-in-Decoder (large)	<b>51.4</b>	<b>67.6</b>	<b>80.1</b>	<b>56.7</b>	63.2



- **ATLAS** [Izacard et al., 2022]: Training objective

- Then, Atlas jointly trains Contriever and FiD, similar to REALM

- Remark. Same decomposition is considered, but different retrieval modeling  $p(z|x)$

$$p(y|x) = \sum_{z \in \mathcal{Z}} p(y|z, x) p(z|x)$$

- Retrieval modeling: Leave-one-out Perplexity Distillation (**LOOP**)

- **Idea**: how much worse the prediction, when removing one of top-k documents

$$p_{\text{LOOP}}(\mathbf{d}_k) = \frac{\exp(-\log p_{LM}(\mathbf{a} | \mathcal{D}_K \setminus \{\mathbf{d}_k\}, \mathbf{q}))}{\sum_{i=1}^K \exp(-\log p_{LM}(\mathbf{a} | \mathcal{D}_K \setminus \{\mathbf{d}_i\}, \mathbf{q}))}$$

- With LOOP, both Contriver and FiD are fine-tuned using masked language modeling

- They are further fine-tuned to solve specific downstream task, *e.g.*, Open-domain QA

# Building Blocks of Foundational Language Models: Retrieval Augment

- **ATLAS** [Izacard et al., 2022]: Experiments

- Comparison to state-of-the-art on **question answering**

- Remark. GPT-3, Gopher, Chinchilla uses prompting, but ATLAS uses fine-tuning for few-shot
- ATLAS outperforms both **LLMs without retrieval** and **existing retrieval-augmented LMs**

Model	NQ		TriviaQA filtered		TriviaQA unfiltered	
	64-shot	Full	64-shot	Full	64-shot	Full
GPT-3 (Brown et al., 2020)	29.9	-	-	-	71.2	-
Gopher (Rae et al., 2021)	28.2	-	57.2	-	61.3	-
Chinchilla (Hoffmann et al., 2022)	35.5	-	64.6	-	72.3	-
PaLM (Chowdhery et al., 2022)	39.6	-	-	-	81.4	-
RETRO (Borgeaud et al., 2021)	-	45.5	-	-	-	-
FiD (Izacard & Grave, 2020)	-	51.4	-	67.6	-	80.1
FiD-KD (Izacard & Grave, 2021)	-	54.7	-	73.3	-	-
R2-D2 (Fajcik et al., 2021)	-	55.9	-	69.9	-	-
ATLAS	<b>42.4</b>	<b>60.4</b>	<b>74.5</b>	<b>79.8</b>	<b>84.7</b>	<b>89.4</b>

# Building Blocks of Foundational Language Models: Retrieval Augment

- **ATLAS** [Izacard et al., 2022]: Experiments

- Comparison to state-of-the-art on **MMLU** (57 tasks)

- Remark. GPT-3, Gopher, Chinchilla uses prompting, but ATLAS uses fine-tuning for few-shot
- For 5-shot setup, ATLAS outperforms GPT-3 with 16 times smaller parameters
- With full training, ATLAS even can outperform stronger LLMs such as Gopher

Setting	Model	Params	Train FLOPS	All	Hum.	Soc. Sci.	STEM	Other
zero-shot	ATLAS	11B	3.5e22	47.1	43.6	54.1	38.0	54.4
5-shot	GPT-3	175B	3.1e23	43.9	40.8	50.4	36.7	48.8
	Gopher	280B	5.0e23	60.0	56.2	71.9	47.4	66.1
	Chinchilla	70B	5.0e23	<b>67.5</b>	<b>63.6</b>	<b>79.3</b>	<b>55.0</b>	73.9
	ATLAS*	11B	3.5e22	47.9	46.1	54.6	38.8	52.8
5-shot (multi-task)	ATLAS	11B	3.5e22	56.6	50.1	66.4	46.4	66.2
Full / Transfer	UnifiedQA	11B	3.3e22	48.9	45.6	56.6	40.2	54.6
	GPT-3	175B	3.1e23	53.9	52.5	63.9	41.4	57.9
	ATLAS	11B	3.5e22	66.0	61.1	77.2	53.2	<b>74.4</b>



## 1. Preliminary

- Important properties of large language models
- Large language models beyond GPT-3

## 2. Building Blocks of Large Language Models

- Prompt-tuning
- Alignment with human values and intent
- Retrieval augmentation

## 3. Recent Advances of Large Language Models

- LLMs as optimizers
- Tool use
- Self improvement
- AI agents

# Recent Advances of Large Language Models: LLMs as Optimizers

- **OPRO** [Yang et al., 2024]
  - LLMs as Optimizers
    - By utilizing in-context learning power of LLM, LLM can solve real-world optimization problem that are absence of gradient. (e.g. prompt optimization)

I have some texts along with their corresponding scores. The texts are arranged in ascending order based on their scores, where higher scores indicate better quality.

text:  
Let's figure it out!  
score:  
61

text:  
Let's solve the problem.  
score:  
63

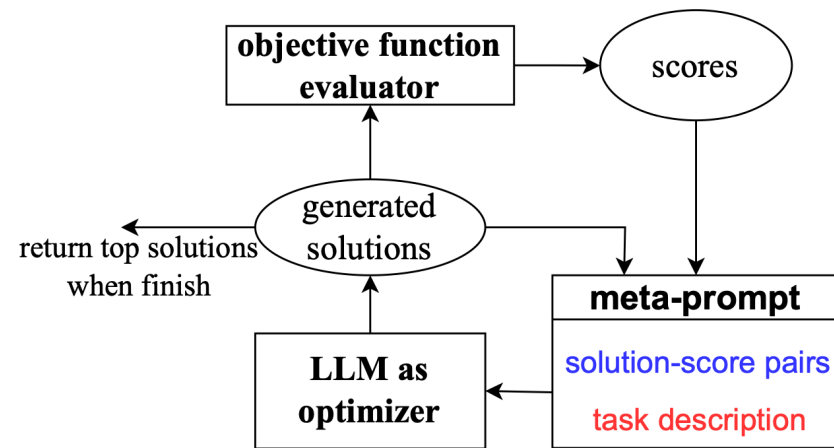
(... more instructions and scores ...)

The following exemplars show how to apply your text: you replace <INS> in each input with your text, then read the input and give an output. We say your output is wrong if your output is different from the given output, and we say your output is correct if they are the same.

input:  
Q: Alannah, Beatrix, and Queen are preparing for the new school year and have been given books by their parents. Alannah has 20 more books than Beatrix. Queen has 1/5 times more books than Alannah. If Beatrix has 30 books, how many books do the three have together?  
A: <INS>  
output:  
140

(... more exemplars ...)

Write your new text that is different from the old ones and has a score as high as possible. Write the text in square brackets.



# Recent Advances of Large Language Models: LLMs as Optimizers

- **OPRO** [Yang et al., 2024]

- Describe the optimization problem in natural language

The following exemplars show how to apply your text: you replace <INS> in each input with your text, then read the input and give an output. We say your output is wrong if your output is different from the given output, and we say your output is correct if they are the same.

input:

Q: Alannah, Beatrix, and Queen are preparing for the new school year and have been given books by their parents. Alannah has 20 more books than Beatrix. Queen has 1/5 times more books than Alannah. If Beatrix has 30 books, how many books do the three have together?

A: <INS>

output:

140

(... more exemplars ...)

Write your new text that is different from the old ones and has a score as high as possible. Write the text in square brackets.

- Instruct the LLM to iteratively generate new solutions based on the problem description and the previously found solutions

I have some texts along with their corresponding scores. The texts are arranged in ascending order based on their scores, where higher scores indicate better quality.

text:

Let's figure it out!

score:

61

text:

Let's solve the problem.

score:

63

(... more instructions and scores ...)

Evaluation Score



- **OPRO** [Yang et al., 2024]: Experiments

- The most simple optimization problem: ***Linear Regression***

- The goal is to find the linear coefficients that probabilistically best explain the response from the input variables
    - Predict  $w_{\text{true}}$  and  $b_{\text{true}}$

$$y = w_{\text{true}}x + b_{\text{true}} + \epsilon$$

# Recent Advances of Large Language Models: LLMs as Optimizers

- **OPRO** [Yang et al., 2024]: Experiments

- The most simple optimization problem: **Linear Regression**

- The goal is to find the linear coefficients that probabilistically best explain the response from the input variables
- Predict  $w_{\text{true}}$  and  $b_{\text{true}}$

$$y = w_{\text{true}}x + b_{\text{true}} + \epsilon$$

$w_{\text{true}}$	$b_{\text{true}}$	number of steps			number of unique $(w, b)$ pairs explored		
		text-bison	gpt-3.5-turbo	gpt-4	text-bison	gpt-3.5-turbo	gpt-4
15	14	5.8 ± 2.6	7.6 ± 4.5	<b>4.0</b> ± 1.5	40.0 ± 12.4	36.0 ± 15.2	<b>17.2</b> ± 5.1
17	17	<b>4.0</b> ± 1.8	12.6 ± 6.0	6.0 ± 3.7	33.4 ± 11.7	53.8 ± 16.9	<b>26.0</b> ± 10.6
16	10	<b>3.8</b> ± 2.2	10.4 ± 5.4	6.2 ± 3.1	30.2 ± 13.4	42.8 ± 16.3	<b>24.2</b> ± 8.2
3	5	<b>9.8</b> ± 2.8	10.8 ± 2.7	12.2 ± 2.0	55.8 ± 16.1	39.6 ± 10.1	<b>33.0</b> ± 4.0
25	23	19.6 ± 11.4	26.4 ± 18.3	<b>12.2</b> ± 3.7	104.0 ± 52.3	78.6 ± 26.2	<b>44.2</b> ± 8.3
2	30	<b>31.4</b> ± 6.3	42.8 ± 9.7	38.0 ± 15.9	126.4 ± 17.7	125.6 ± 21.7	<b>99.0</b> ± 24.6
36	-1	<b>35.8</b> ± 6.4	45.4 ± 16.9	50.4 ± 18.8	174.0 ± 28.2	142.2 ± 31.2	<b>116.4</b> ± 32.7

- The number of unique explored by each model is fewer than exhaustive search.
- Text-bison and gpt-4 models outperform gpt-3.5-turbo.
- The problem becomes harder for all models when the GT moves farther from the starting region:  $(w,b) \in [10,20] \times [10,20]$

- **OPRO** [Yang et al., 2024]: Experiments
  - Another simple optimization problem: ***Traveling Salesman Problem (TSP)***
    - When given a set of  $n$  nodes with their coordinates, to find the shortest route that traverses all nodes from the starting node and finally returns to the starting node.
  - LLMs starts from 5 randomly generated solutions, and each optimization step produces at most 8 new solutions.
  - Heuristic baselines: Nearest Neighbor (NN), Farthest Insertion (FI)

# Recent Advances of Large Language Models: LLMs as Optimizers

- **OPRO** [Yang et al., 2024]: Experiments

- Another simple optimization problem: **Traveling Salesman Problem (TSP)**
  - When given a set of  $n$  nodes with their coordinates, to find the shortest route that traverses all nodes from the starting node and finally returns to the starting node.
- LLMs starts from 5 randomly generated solutions, and each optimization step produces at most 8 new solutions.
- Heuristic baselines: Nearest Neighbor (NN), Farthest Insertion (FI)

$n$	optimality gap (%)					# steps (# successes)		
	NN	FI	text-bison	gpt-3.5-turbo	gpt-4	text-bison	gpt-3.5-turbo	gpt-4
10	13.0 ± 1.3	3.2 ± 1.4	<b>0.0</b> ± 0.0	<b>0.0</b> ± 0.0	<b>0.0</b> ± 0.0	40.4 ± 5.6 (5)	46.8 ± 9.3 (5)	<b>9.6</b> ± 3.0 (5)
15	9.4 ± 3.7	1.2 ± 0.6	4.4 ± 1.3	1.2 ± 1.1	<b>0.2</b> ± 0.2	N/A (0)	202.0 ± 41.1 (4)	<b>58.5</b> ± 29.0 (4)
20	16.0 ± 3.9	<b>0.2</b> ± 0.1	30.4 ± 10.6	4.4 ± 2.5	1.4 ± 0.6	N/A (0)	438.0 ± 0.0 (1)	<b>195.5</b> ± 127.6 (2)
50	19.7 ± 3.1	<b>9.8</b> ± 1.5	219.8 ± 13.7	133.0 ± 6.8	11.0 ± 2.6	N/A (0)	N/A (0)	N/A (0)

- GPT-4 significantly outperforms other LLMs.
- Limitation: OPRO degrades dramatically on problems with larger sizes.
  - When  $n=10$ , all LLMs find the optimal solutions for every problem: but as the problem size get larger, optimality gaps increase quickly, and FI heuristic starts to outperform all LLMs.

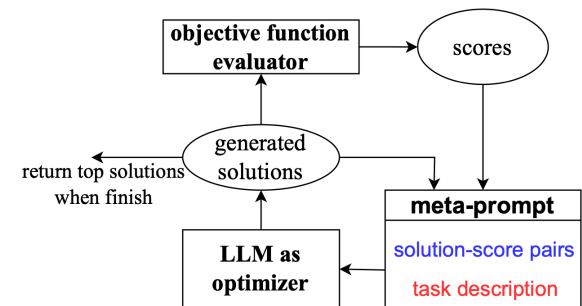
# Recent Advances of Large Language Models: LLMs as Optimizers

- **OPRO** [Yang et al., 2024]: Applications

- Prompt Optimization, where both the input and output are in the text format.
- The output of the optimizer LLM is an *instruction* which is concatenated to the question part of every exemplar and prompts the scorer LLM.
  - *Q\_begin*: the instruction is added before the original question.
  - *Q\_end*: the instruction is added after the original question.
  - *A\_begin*: the instruction is added to the beginning of the scorer LLM output. This is applicable to pretrained LLMs without instruction tuning, where the prompt is formatted as a sequence of QA pairs.

Process:

1. Generate initial solutions
2. Make *meta-prompt* construct with solution-score pairs and task description.
3. Generate new solutions with *optimizer LLM*.
4. Score generated solutions with *scorer LLM*.
5. Iterate {2-4}

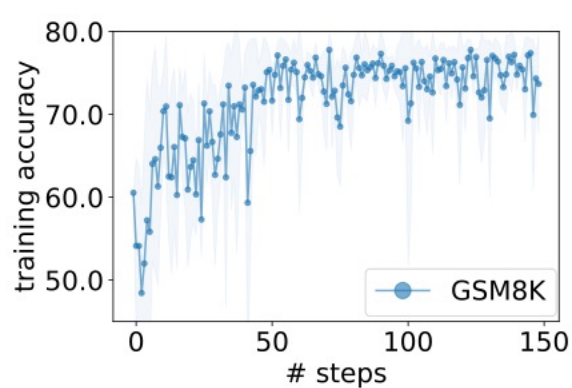




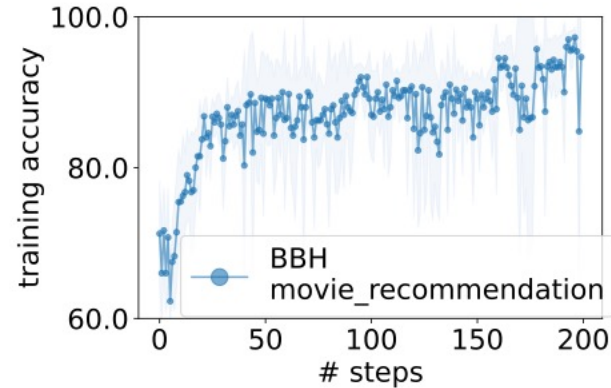
# Recent Advances of Large Language Models: LLMs as Optimizers

- **OPRO** [Yang et al., 2024]: Applications

- Prompt optimization for GSM 8K benchmark (math domain) and BBH benchmark.



(a) GSM8K



(b) BBH movie\_recommendation

- Generated Solutions

- GSM 8K: "Take a deep breath and work on this problem step-by-step." Score: 80.2
- BBH: "Based on your input, I have analyzed the given movies in terms of genre, plot, tone, audience rating, year of release, director, cast, and reviews. I have also taken into account the given options. The movie that is most similar to the given movies in terms of all these factors is:", Score: 90.8

- **OPRO** [Yang et al., 2024]: Conclusion
  - Use LLM as optimizer to solve optimization problem.
    - LLM is effective optimizer for optimization problems: Experiments demonstrate that LLM has the ability to progressively improve solutions generated based on optimization trajectories.
  - Prompt optimization with OPRO significantly improves task performance.
    - In GSM8K and BBH benchmarks, outperform human-designed prompts, and in some cases performance gains of up to 50% or more.
  - OPRO has some open challenges.
    - Important to reduce sensitivity to initialization
    - Improve how to balance between exploitation and exploration.
  - OPRO can be utilized to new tasks beyond prompt optimization.

# Recent Advances of Large Language Models: Tool-use

- **PAL** [Gao et al., 2023]
  - **Motivation:** natural language might be not optimal way to solve given task
    - E.g., mathematical reasoning → *do LLMs really know how to add/multiplication?*

## 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.



## Does sparks of AGI (i.e., GPT4) solve multiplication?



GPT4 achieves only **59% accuracy** on 3-digit x 3-digit multiplication.

**4% accuracy for 4x4 multiplication!**

**0% accuracy for 5x5 multiplication!**

# Recent Advances of Large Language Models: Tool-use

- **PAL** [Gao et al., 2023]
  - **Motivation**: natural language might be not optimal way to solve given task
  - **Solution**: let LLMs utilize the external tool for given task, *e.g.*, calculator or python
    - Idea: generating both **language rationale** (similar to CoT) and **python code** together
    - Then, **final answer** is obtained by executing codes (language part will be *#comment*)

## 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. The answer is
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
```



# Recent Advances of Large Language Models: Tool-use

- **PAL** [Gao et al., 2023]
  - **Motivation**: natural language might be not optimal way to solve given task
  - **Solution**: let LLMs utilize the external tool for given task, *e.g.*, calculator or python
  - **More examples** of prompt
    - Mathematical reasoning

Q: Olivia has \$23. She bought five bagels for \$3 each. How much money does she have left?

```
money_initial = 23
bagels = 5
bagel_cost = 3
money_spent = bagels * bagel_cost
money_left = money_initial - money_spent
answer = money_left
```

- Symbolic reasoning: Colored objects

Q: On the table, you see a bunch of objects arranged in a row: a purple paperclip, a pink stress ball, a brown keychain, a green scrunchiephone charger, a mauve fidget spinner, and a burgundy pen. What is the color of the object directly to the right of the stress ball?

```
...
stress_ball_idx = None
for i, object in enumerate(objects):
    if object[0] == 'stress ball':
        stress_ball_idx = i
        break
# Find the directly right object
direct_right = objects[stress_ball_idx+1]
# Check the directly right object's color
answer = direct_right[1]
```

# Recent Advances of Large Language Models: Tool-use

- **PAL** [Gao et al., 2023]: Experiments

- Solve rate (%) on mathematical reasoning tasks

	GSM8K	GSM-HARD	SVAMP	ASDIV	SINGLEEQ	SINGLEOP	ADDSUB	MULTIARITH
DIRECT <sub>Codex</sub>	19.7	5.0	69.9	74.0	86.8	93.1	90.9	44.0
COT <sub>UL2-20B</sub>	4.1	-	12.6	16.9	-	-	18.2	10.7
COT <sub>LaMDA-137B</sub>	17.1	-	39.9	49.0	-	-	52.9	51.8
COT <sub>Codex</sub>	65.6	23.1	74.8	76.9	89.1	91.9	86.0	95.9
COT <sub>PaLM-540B</sub>	56.9	-	79.0	73.9	92.3	94.1	91.9	94.7
COT <sub>Minerva 540B</sub>	58.8	-	-	-	-	-	-	-
<b>PAL</b>	<b>72.0</b>	<b>61.2</b>	<b>79.4</b>	<b>79.6</b>	<b>96.1</b>	<b>94.6</b>	<b>92.5</b>	<b>99.2</b>

- Solve rate (%) on symbolic reasoning tasks

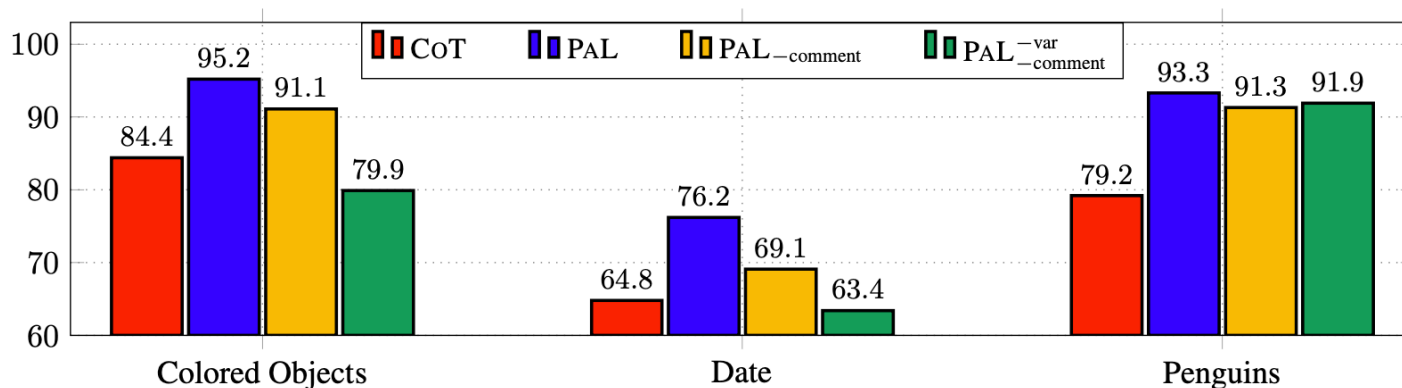
	COLORED OBJECT	PENGUINS	DATE	REPEAT COPY	OBJECT COUNTING
DIRECT <sub>Codex</sub>	75.7	71.1	49.9	81.3	37.6
COT <sub>LaMDA-137B</sub>	-	-	26.8	-	-
COT <sub>PaLM-540B</sub>	-	65.1	65.3	-	-
COT <sub>Codex</sub>	86.3	79.2	64.8	68.8	73.0
<b>PAL<sub>Codex</sub></b>	<b>95.1</b>	<b>93.3</b>	<b>76.2</b>	<b>90.6</b>	<b>96.7</b>

# Recent Advances of Large Language Models: Tool-use

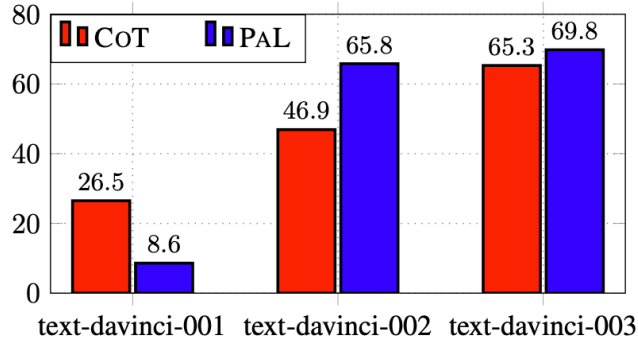
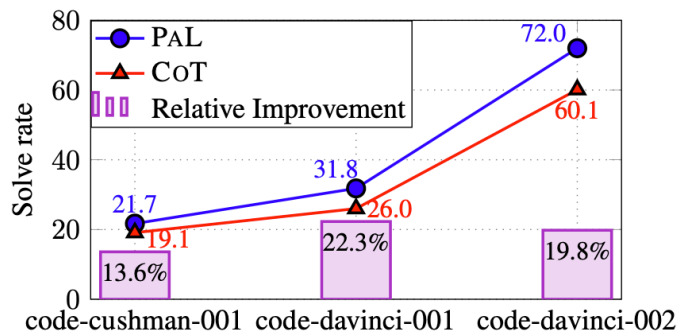
- **PAL** [Gao et al., 2023]: Experiments

- Ablation studies

- Including language rationale as comment is positive for accuracy (blue > yellow)
- Naming variable with relevant functionality is very important (blue > green)

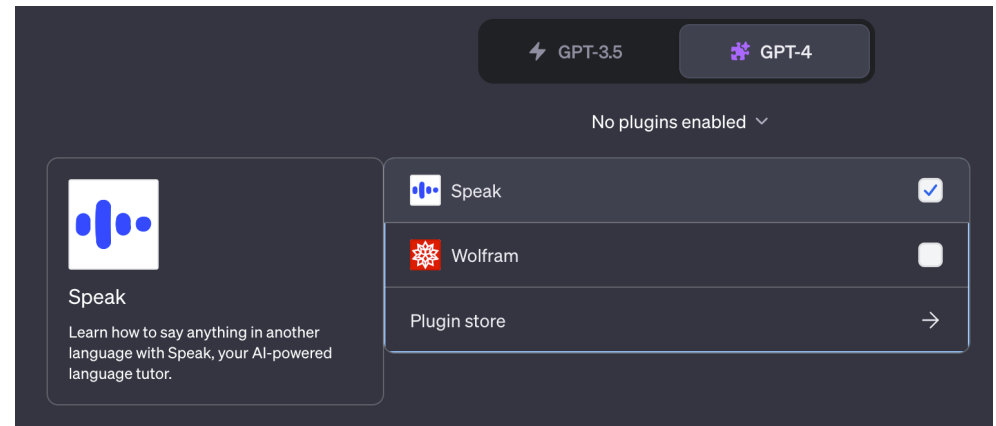
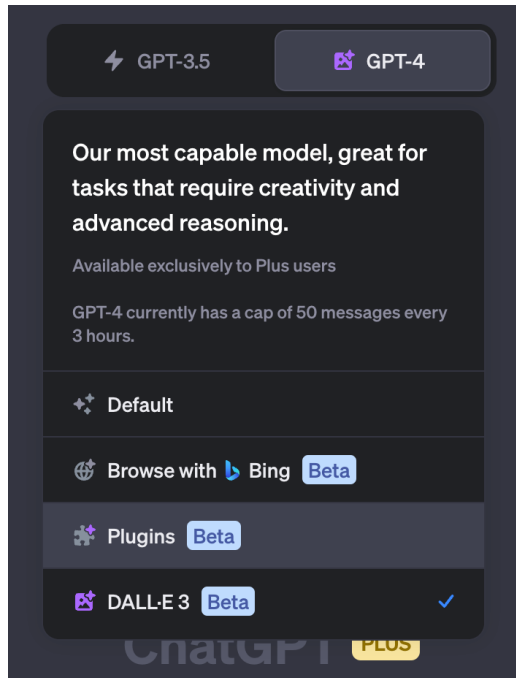


- Generalization with different sizes and LLMs (on GSM8K)



# Recent Advances of Large Language Models: Tool-use

- **PAL** [Gao et al., 2023]: Applications
  - This feature is closely related with ChatGPT's plugin
    - Similar intuition with PAL while it's not open (from instruction-tuning or prompting)

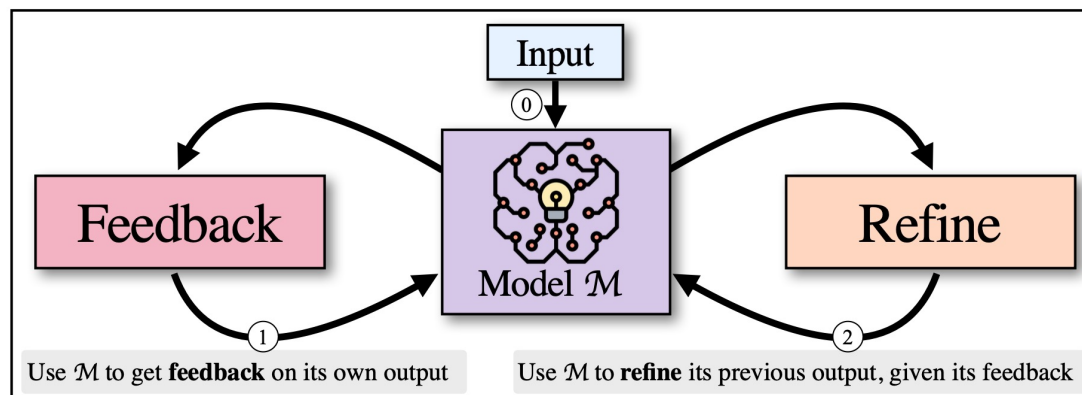




# Recent Advances of Large Language Models: Self-feedback

- **Self-refine** [Madaan et al., 2023]

- More interestingly, foundation model can give feedback and refine itself
  - *i.e.*, both feedback & refine are conducted from foundation model with **different prompt**



---

## Algorithm 1 SELF-REFINE algorithm

---

**Require:** input  $x$ , model  $\mathcal{M}$ , prompts  $\{p_{\text{gen}}, p_{\text{fb}}, p_{\text{refine}}\}$ , stop condition  $\text{stop}(\cdot)$

```
1:  $y_0 = \mathcal{M}(p_{\text{gen}} \| x)$  ▷ Initial generation (Eqn. 1)
2: for iteration  $t \in 0, 1, \dots$  do
3:    $fb_t = \mathcal{M}(p_{\text{fb}} \| x \| y_t)$  ▷ Feedback (Eqn. 2)
4:   if  $\text{stop}(fb_t, t)$  then ▷ Stop condition
5:     break
6:   else
7:      $y_{t+1} = \mathcal{M}(p_{\text{refine}} \| x \| y_0 \| fb_0 \| \dots \| y_t \| fb_t)$  ▷ Refine (Eqn. 4)
8:   end if
9: end for
10: return  $y_t$ 
```

---

# Recent Advances of Large Language Models: Self-feedback

---

- **Self-refine** [Madaan et al., 2023]
  - More interestingly, foundation model can give feedback and refine itself
  - Example: code optimization (single iteration)
    1. Initial generation

Input:

```
Question - Find the square root of a number.
```

Initial Generation

Output:

```
a, b = input().split()
n = int(a + b)

flag = False
for i in range(n):
    if i ** 2 == n:
        flag = True
        break

print('Yes' if flag else 'No')
```

# Recent Advances of Large Language Models: Self-feedback

- **Self-refine** [Madaan et al., 2023]
  - More interestingly, foundation model can give feedback and refine itself
  - Example: code optimization (single iteration)

## 2. Feedback

Question - Find the square root of a number.

### Initial Generation

Input:

```
a, b = input().split()
n = int(a + b)

flag = False
for i in range(n):
    if i ** 2 == n:
        flag = True
        break

print('Yes' if flag else 'No')
```

### Feedback

```
Feedback:
# Why is this code slow?
```

Output:

```
# This code is slow because it is using a brute force approach to find the square |
```

# Recent Advances of Large Language Models: Self-feedback

---

- **Self-refine** [Madaan et al., 2023]
  - More interestingly, foundation model can give feedback and refine itself
  - Example: code optimization (single iteration)

## 3. Refinement

⋮

Input:

```
Feedback:  
# Why is this code slow?  
  
# This code is slow because it is using a brute force approach to find the square |
```

Refinement

Output:

```
# Improved version:  
  
a, b = input().split()  
n = int(a + b)  
  
flag = False  
for i in range(1000):  
    if i ** 2 == n:  
        flag = True  
        break  
  
print('Yes' if flag else 'No')
```

# Recent Advances of Large Language Models: Self-feedback

- **Self-refine** [Madaan et al., 2023]: Experiments

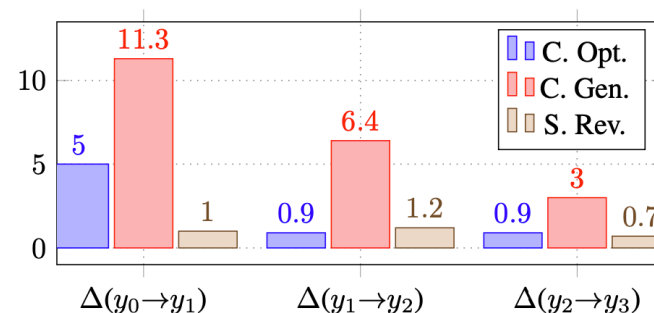
- Overall results

- This framework is well generalized across different LLMs
- Remark. For each task, specific metric is used, *e.g.*, accuracy or human preference

Task	GPT-3.5		ChatGPT		GPT-4	
	Base	+SELF-REFINE	Base	+SELF-REFINE	Base	+SELF-REFINE
Sentiment Reversal	8.8	<b>30.4</b> (↑21.6)	11.4	<b>43.2</b> (↑31.8)	3.8	<b>36.2</b> (↑32.4)
Dialogue Response	36.4	<b>63.6</b> (↑27.2)	40.1	<b>59.9</b> (↑19.8)	25.4	<b>74.6</b> (↑49.2)
Code Optimization	14.8	<b>23.0</b> (↑8.2)	23.9	<b>27.5</b> (↑3.6)	27.3	<b>36.0</b> (↑8.7)
Code Readability	37.4	<b>51.3</b> (↑13.9)	27.7	<b>63.1</b> (↑35.4)	27.4	<b>56.2</b> (↑28.8)
Math Reasoning	<b>64.1</b>	<b>64.1</b> (0)	74.8	<b>75.0</b> (↑0.2)	92.9	<b>93.1</b> (↑0.2)
Acronym Generation	41.6	<b>56.4</b> (↑14.8)	27.2	<b>37.2</b> (↑10.0)	30.4	<b>56.0</b> (↑25.6)
Constrained Generation	28.0	<b>37.0</b> (↑9.0)	44.0	<b>67.0</b> (↑23.0)	15.0	<b>45.0</b> (↑30.0)

- Iteration-wise score improvement

Task	$y_0$	$y_1$	$y_2$	$y_3$
Code Opt.	22.0	27.0	27.9	<b>28.8</b>
Sentiment Rev.	33.9	34.9	36.1	<b>36.8</b>
Constrained Gen.	29.0	40.3	46.7	<b>49.7</b>



# Recent Advances of Large Language Models: Self-Refine

- **Self-refine** [Madaan et al., 2023]: Application
- **MCT Self-Refine** [zhang et al., 2024]
  - Amplify LLM's mathematical reasoning ability by constructing MonteCarlo search tree through iterative process of Selection, self-refine, self-evaluation, and Backpropagation.
  - They showed that they can enhance mathematical reasoning ability even for the small model (**LLaMa3-8B**)

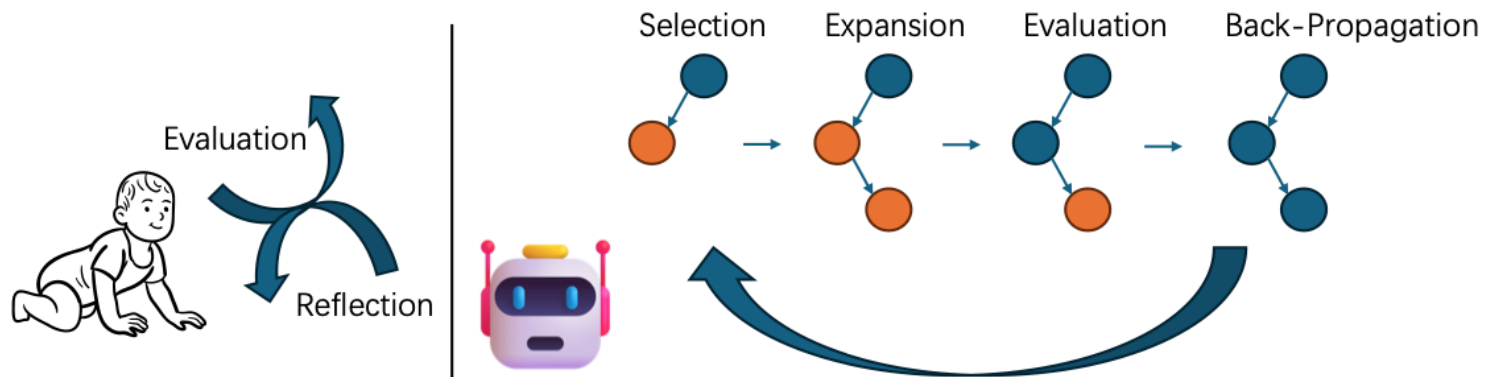


Figure 1: Agents can learn decision-making and reasoning from the trial-and-error as humans do.

- **MCT Self-Refine** [zhang et al., 2024]
  - MonteCarlo Tree Search
    - **Initialization:** A root node is established using either a naive model-generated answer and a dummy response (e.g., 'I don't know.')
    - **Selection:** Selects the highest-valued node based on value function  $Q$  for further exploration and refinement using a greedy strategy.
    - **Self-Refine:** The selected answer undergoes optimization using the **Self-Refine framework (Madaan et al., 2023)**.
    - **Self-Evaluation:** The refined answer is scored to sample a reward value and compute its  $Q$  value. This involves model self-reward feedback and constraints such as strict scoring standards and suppression of perfect scores to ensure reliability and fairness in scoring.
    - **Backpropagation:** The value of the refined answer is propagated backward to its parent node and other related nodes to update the tree's value information. If the  $Q$  value of any child node changes, the parent node's  $Q$  is updated.
    - **UCT update:** After the  $Q$  values of all nodes are updated, we identify a collection  $C$  of candidate nodes for further expansion or Selection, then use the UCT update formula to update the UCT values of all nodes for the next Selection stage.

# Recent Advances of Large Language Models: Self-Refine

- **MCT Self-Refine** [zhang et al., 2024]: Experiments
  - GSM Benchmark and MATH Benchmark

Datasets	Zero-Shot CoT	One-turn Self-refine	4-rollouts MCTSr	8-rollouts MCTSr	Example Nums
GSM8K	977	1147	1227	1275	1319
	74.07%	86.96%	93.03%	96.66%	
GSM-Hard	336	440	526	600	1319
	25.47%	33.36%	39.88%	45.49%	

Table 1: Performance of MCTSr on the GSM Dataset

Level	Zero-Shot CoT	One-turn Self-refine	4-rollouts MCTSr	8-rollouts MCTSr	Example Nums
level-1	250	314	365	394	437
	57.21%	71.85%	83.52%	90.16%	
level-2	363	474	594	692	894
	40.60%	53.02%	66.44%	77.40%	
level-3	309	454	585	719	1131
	27.32%	40.14%	51.72%	63.57%	
level-4	202	368	523	656	1214
	16.64%	30.31%	43.08%	54.04%	
level-5	94	177	290	451	1324
	7.10%	13.37%	21.90%	34.06%	
Overall	1218	1787	2357	2912	5000
	24.36%	35.74%	47.14%	58.24%	

Table 2: Performance of MCTSr on the MATH Dataset

- MCTSr algorithm's potential in academic and problem-solving contexts.



# Recent Advances of Large Language Models: Self-Refine

- **MCT Self-Refine** [zhang et al., 2024]: Experiments
  - Olympiad level Benchmarks

Datasets	Zero-Shot CoT	One-turn Self-refine	4-rollouts MCTSr	8-rollouts MCTSr	Example Nums
AIME	22	41	70	110	933
	2.36%	4.39%	7.50%	11.79%	
Math Odyssey	67	118	156	192	389
	17.22%	30.33%	40.10%	49.36%	
OlympiadBench	16	39	67	99	1275
	1.25%	3.06%	5.25%	7.76%	

Table 3: Performance of MCTSr on Olympiad-level Datasets

- Comparison with SOTA closed LLMs
  - MCTSr can effectively enhance the mathematical reasoning capabilities of small-parameter open-source models, like LLaMa-3, to a comparable level.

	Gemini 1.5-Pro	Claude 3 Opus	GPT-4 Turbo
MATH (Reid et al., 2024)	67.7	60.1	73.4
Math Odyssey (Reid et al., 2024)	45.0	40.	49.1
GSM8K (Papers with Code, 2024)	94.4	95	97.1

Table 4: closed-source LLM performance on mathematical datasets

# Recent Advances of Large Language Models: AI Agents

- **Generative agents** [Park et al., 2023]
  - Then, what if we simulate human behavior using LLMs?
    - Using the history of each human as prompt and allowing actions on environment (Sims)



# Recent Advances of Large Language Models: AI Agents

- **Generative agents** [Park et al., 2023]
  - Then, what if we simulate human behavior using LLMs?
    - Using the history of each human as prompt and allowing actions on environment (Sims)



# Recent Advances of Large Language Models: AI Agents

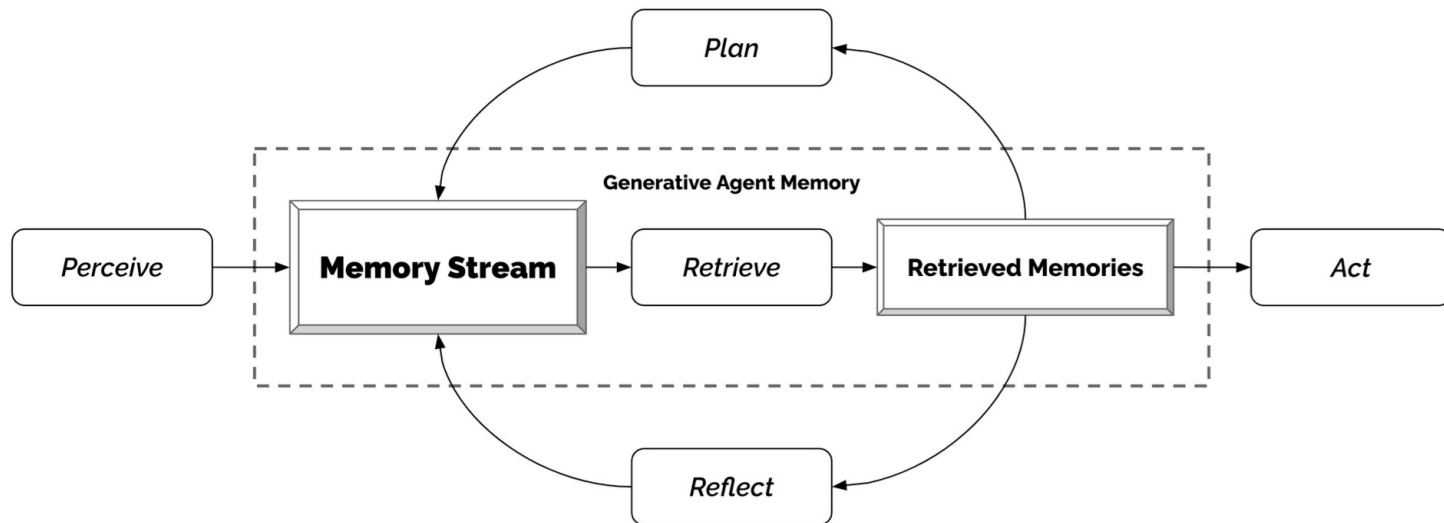
- **Generative agents** [Park et al., 2023]

- Then, what if we simulate human behavior using LLMs?
- Interestingly, each character powered by LLMs show many different behavior depending on given characteristics similar to human



- **Generative agents** [Park et al., 2023]: **Overview**

- Goal: interaction with other agents and react to changes in environment
- Method: agent architecture combining LLMs with novel mechanisms such that synthesizing/retrieving relevant information to condition LLMs' output
  - Key feature: **Memory stream**



# Recent Advances of Large Language Models: AI Agents

- **Generative agents** [Park et al., 2023]: Technical Details – **Memory/Retrieval**
  - **Target challenge.** Not all experience is essential & limited context window of LLMs
  - **Solution:** retrieving relevant experience from memory stream of observations
    - *Observation*; event directly perceived by agent (time stamp + language description)

**Memory Stream**

```
2023-02-13 22:48:20: desk is idle
2023-02-13 22:48:20: bed is idle
2023-02-13 22:48:10: closet is idle
2023-02-13 22:48:10: refrigerator is idle
2023-02-13 22:48:10: Isabella Rodriguez is stretching
2023-02-13 22:33:30: shelf is idle
2023-02-13 22:33:30: desk is neat and organized
2023-02-13 22:33:10: Isabella Rodriguez is writing in her journal
2023-02-13 22:18:10: desk is idle
2023-02-13 22:18:10: Isabella Rodriguez is taking a break
2023-02-13 21:49:00: bed is idle
2023-02-13 21:48:50: Isabella Rodriguez is cleaning up the kitchen
2023-02-13 21:48:50: refrigerator is idle
2023-02-13 21:48:50: bed is being used
2023-02-13 21:48:10: shelf is idle
2023-02-13 21:48:10: Isabella Rodriguez is watching a movie
2023-02-13 21:19:10: shelf is organized and tidy
2023-02-13 21:18:10: desk is idle
2023-02-13 21:18:10: Isabella Rodriguez is reading a book
2023-02-13 21:03:40: bed is idle
2023-02-13 21:03:30: refrigerator is idle
2023-02-13 21:03:30: desk is in use with a laptop and some papers on it
...
```

**Q. What are you looking forward to the most right now?**

Isabella Rodriguez is excited to be planning a Valentine's Day party at Hobbs Cafe on February 14th from 5pm and is eager to invite everyone to attend the party.

retrieval	=	recency	+	importance	+	relevance
2.34	=	0.91	+	0.63	+	0.80

ordering decorations for the party

2.21	=	0.87	+	0.63	+	0.71
------	---	------	---	------	---	------

researching ideas for the party

2.20	=	0.85	+	0.73	+	0.62
------	---	------	---	------	---	------

...

I'm looking forward to the Valentine's Day party that I'm planning at Hobbs Cafe!



# Recent Advances of Large Language Models: AI Agents

- **Generative agents** [Park et al., 2023]: Technical Details – **Memory/Retrieval**
  - **Target challenge.** Not all experience is essential & limited context window of LLMs
  - **Solution:** retrieving relevant experience from memory stream
    - *Retrieval*; consider three features (recency, importance, relevance)
    - Recency: recently happened event has a higher weight

**Memory Stream**

```
2023-02-13 22:48:20: desk is idle
2023-02-13 22:48:20: bed is idle
2023-02-13 22:48:10: closet is idle
2023-02-13 22:48:10: refrigerator is idle
2023-02-13 22:48:10: Isabella Rodriguez is stretching
2023-02-13 22:33:30: shelf is idle
2023-02-13 22:33:30: desk is neat and organized
2023-02-13 22:33:10: Isabella Rodriguez is writing in her journal
2023-02-13 22:18:10: desk is idle
2023-02-13 22:18:10: Isabella Rodriguez is taking a break
2023-02-13 21:49:00: bed is idle
2023-02-13 21:48:50: Isabella Rodriguez is cleaning up the kitchen
2023-02-13 21:48:50: refrigerator is idle
2023-02-13 21:48:50: bed is being used
2023-02-13 21:48:10: shelf is idle
2023-02-13 21:48:10: Isabella Rodriguez is watching a movie
2023-02-13 21:19:10: shelf is organized and tidy
2023-02-13 21:18:10: desk is idle
2023-02-13 21:18:10: Isabella Rodriguez is reading a book
2023-02-13 21:03:40: bed is idle
2023-02-13 21:03:30: refrigerator is idle
2023-02-13 21:03:30: desk is in use with a laptop and some papers on it
...
```

**Q. What are you looking forward to the most right now?**

Isabella Rodriguez is excited to be planning a Valentine's Day party at Hobbs Cafe on February 14th from 5pm and is eager to invite everyone to attend the party.

retrieval	=	recency	+	importance	+	relevance
2.34	=	0.91	+	0.63	+	0.80

ordering decorations for the party

2.21	=	0.87	+	0.63	+	0.71
------	---	------	---	------	---	------

researching ideas for the party

2.20	=	0.85	+	0.73	+	0.62
------	---	------	---	------	---	------

...

I'm looking forward to the Valentine's Day party that I'm planning at Hobbs Cafe!

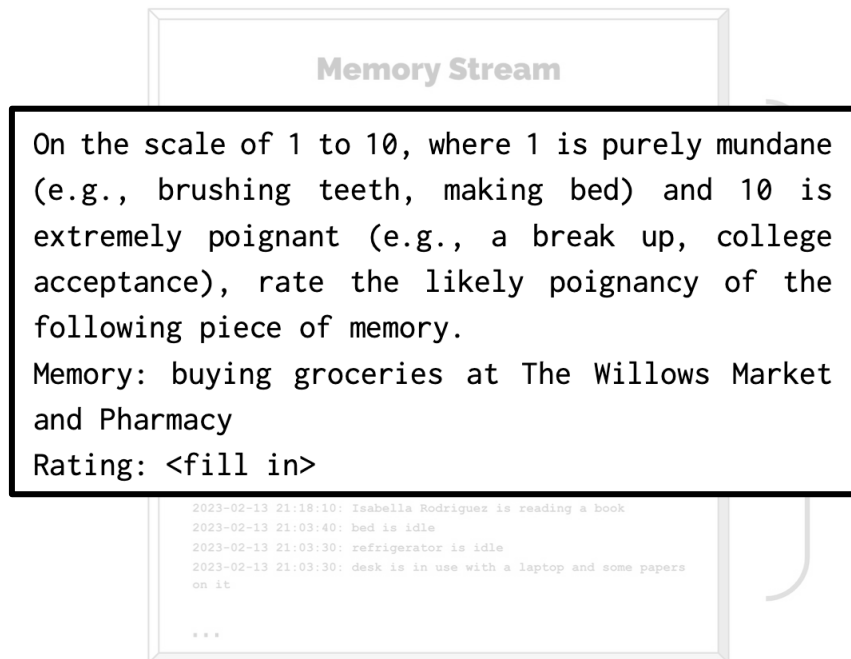


Isabella

# Recent Advances of Large Language Models: AI Agents

- **Generative agents** [Park et al., 2023]: Technical Details – **Memory/Retrieval**

- **Target challenge.** Not all experience is essential & limited context window of LLMs
- **Solution:** retrieving relevant experience from memory stream
  - *Retrieval*; consider three features (*recency*, importance, *relevance*)
  - Importance: rareness of events regardless of given context



## Q. What are you looking forward to the most right now?

Isabella Rodriguez is excited to be planning a Valentine's Day party at Hobbs Cafe on February 14th from 5pm and is eager to invite everyone to attend the party.

retrieval	=	recency	+	importance	+	relevance
2.34	=	0.91	+	0.63	+	0.80

ordering decorations for the party

2.21	=	0.87	+	0.63	+	0.71
------	---	------	---	------	---	------

researching ideas for the party

2.20	=	0.85	+	0.73	+	0.62
------	---	------	---	------	---	------

...

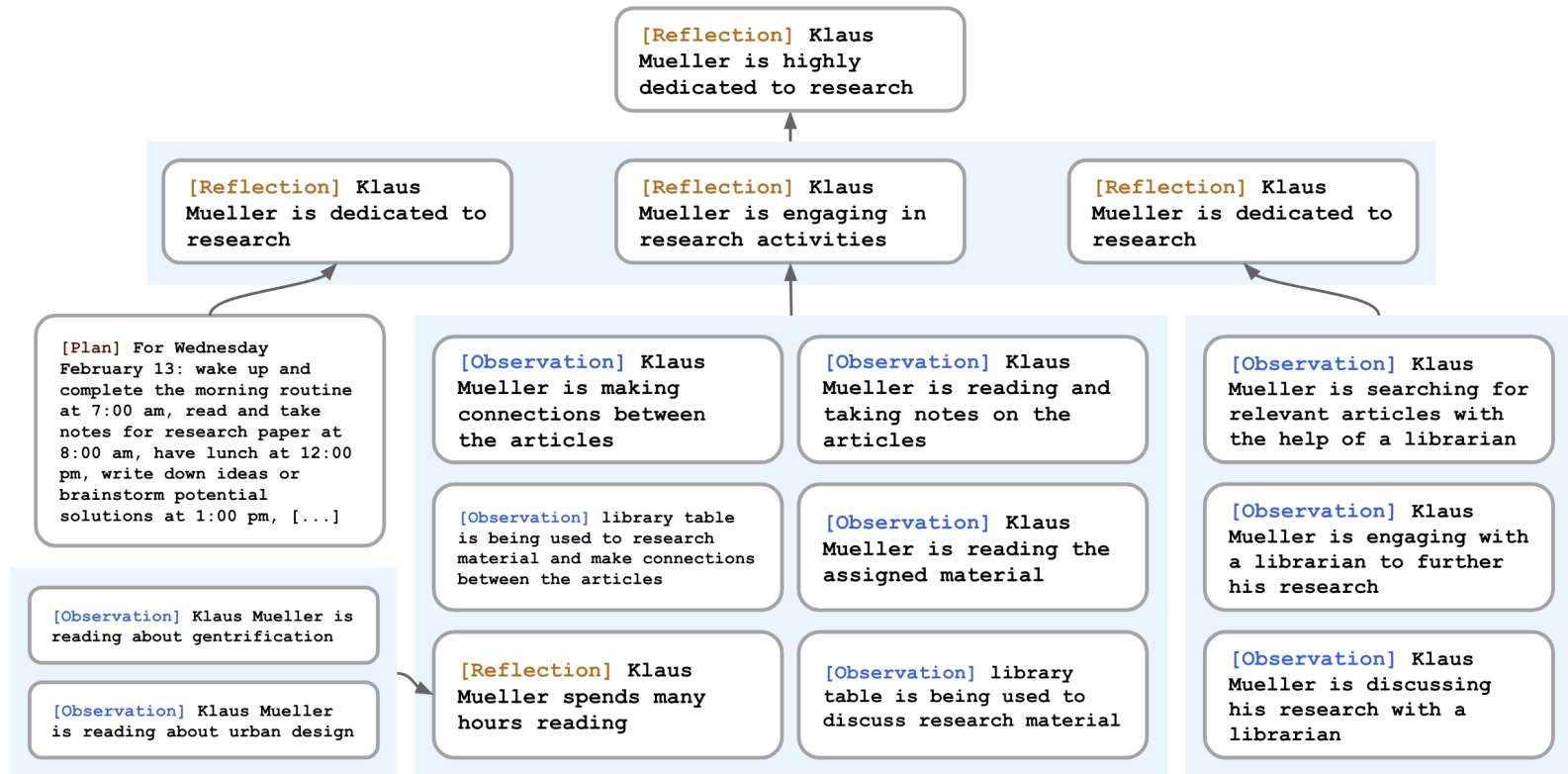
I'm looking forward to the Valentine's Day party that I'm planning at Hobbs Cafe!





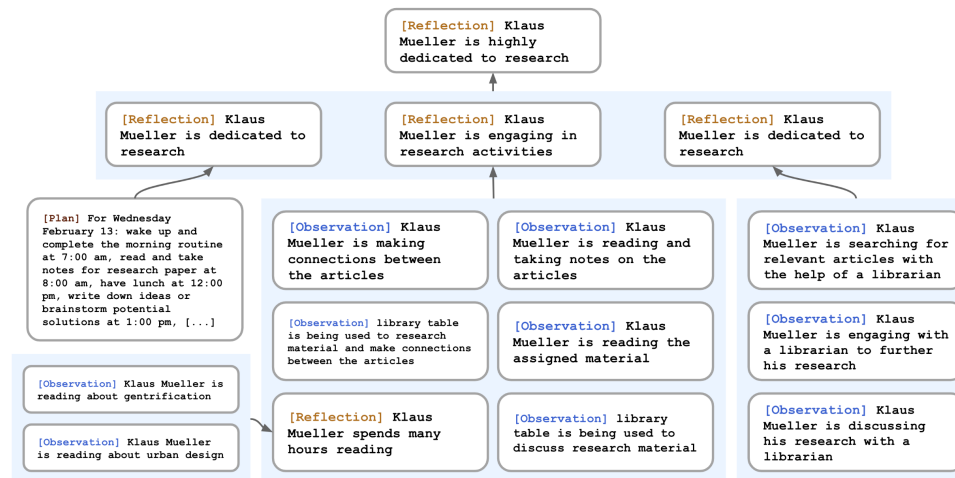
# Recent Advances of Large Language Models: AI Agents

- **Generative agents** [Park et al., 2023]: Technical Details – **Reflection**
  - **Target challenge.** Retrieval is not enough to describe overall status of agent
  - **Solution:** high-level summarization regarding current status of agent
    - *E.g.*, Agent named Klaus Mueller is highly dedicated to research



# Recent Advances of Large Language Models: AI Agents

- **Generative agents** [Park et al., 2023]: Technical Details – **Reflection**
  - **Target challenge.** Retrieval is not enough to describe overall status of agent
  - **Solution:** high-level summarization regarding current status of agent
  - **Step 1.** Prompting to obtain questions to gather high-level information of agent
    - Used prompt: “Given only the information above (100 recent records), what are 3 most salient high-level questions we can answer about the subjects in the statements?”
    - Example responses: “What topic is Klaus Mueller passionate about?” & “What is the relationship between Klaus Mueller and Maria Lopez?”



- **Generative agents** [Park et al., 2023]: Technical Details – **Reflection**

- **Target challenge.** Retrieval is not enough to describe overall status of agent
- **Solution:** high-level summarization regarding current status of agent
- *Step 2.* Gather relevant information for question, then prompting to extract status
  - Query: “What topic is Klaus Mueller passionate about?” (query)
  - Prompt

Statements about Klaus Mueller

1. Klaus Mueller is writing a research paper
2. Klaus Mueller enjoys reading a book on gentrification
3. Klaus Mueller is conversing with Ayesha Khan about exercising [...]

retrieved information

What 5 high-level insights can you infer from the above statements? (example format: insight (because of 1, 5, 3))

- Response: “Klaus Mueller is dedicated to his research on gentrification (because of 1, 2, 8, 15)”

- **Generative agents** [Park et al., 2023]: Technical Details – **Planning and Reacting**
  - **Target challenge.** Ensuring that sequence of actions is coherent and believable
  - **Solution:** Top-down and then recursively generate more detailed plans
    - First, generating overall plans of whole day from agent’s summary description
    - Input prompt:

```
Name: Eddy Lin (age: 19)
Innate traits: friendly, outgoing, hospitable
Eddy Lin is a student at Oak Hill College studying
music theory and composition. He loves to explore
different musical styles and is always looking for
ways to expand his knowledge. Eddy Lin is working
on a composition project for his college class. He
is taking classes to learn more about music theory.
Eddy Lin is excited about the new composition he
is working on but he wants to dedicate more hours
in the day to work on it in the coming days
On Tuesday February 12, Eddy 1) woke up and
completed the morning routine at 7:00 am, [. . .]
6) got ready to sleep around 10 pm.
Today is Wednesday February 13. Here is Eddy’s
plan today in broad strokes: 1)
```

- Output: “1) wake up and complete the morning routine at 8:00 am, 2) go to Oak Hill College to take classes starting 10:00 am, [. . .], 5) work on his new music composition from 1:00 pm to 5:00 pm, 6) have dinner at 5:30 pm, 7) finish school assignments and go to bed by 11:00 pm”

- **Generative agents** [Park et al., 2023]: Technical Details – **Planning and Reacting**
  - **Target challenge.** Ensuring that sequence of actions is coherent and believable
  - **Solution:** Top-down and then recursively generate more detailed plans
    - Then, creating finer-grained actions (day → hours → 5-15 minute chunks)
    - Input prompt: “work on his new music composition from 1:00 pm to 5:00 pm”
    - Output: “1:00 pm: start by brainstorming some ideas for his music composition [...] 4:00 pm: take a quick break and recharge his creative energy before reviewing and polishing his composition.”
  - In addition, these plans could be updated with **reacting**
    - As the environment is updated in real-time

[Agent’s Summary Description]

It is February 13, 2023, 4:56 pm.

John Lin’s status: John is back home early from work.

Observation: John saw Eddy taking a short walk around his workplace.

Summary of relevant context from John’s memory: Eddy Lin is John’s Lin’s son. Eddy Lin has been working on a music composition for his class. Eddy Lin likes to walk around the garden when he is thinking about or listening to music.

Should John react to the observation, and if so, what would be an appropriate reaction?

## Summary

---

- Foundation models in language recently shows tremendous success
  - It is often called large language models (LLMs)
  - By increasing scale, LLMs obtain intriguing properties such as in-context learning
- But, LLMs still have some limitations and they can be mitigated by
  - Carefully designing input prompt
  - Or fine-tuning LLMs to impose alignment
  - Or incorporating external knowledge via retrieval
- Also, recent LLMs show more interesting capability such as
  - Tool-use
  - Self-feedback
  - AI agents

## References

---

[Brown et al., 2020] “Language Models are Few-Shot Learners.” *NeurIPS 2020*

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

[Kaplan et al., 2020] “Scaling Laws for Neural Language Models.” *arXiv preprint*

Link: <https://arxiv.org/abs/2001.08361>

[Wei et al., 2022] “Emergent Abilities of Large Language Models.” *TMLR 2022*

Link: <https://arxiv.org/abs/2206.07682>

[Rae et al., 2021] “Scaling Language Models: Methods, Analysis & Insights from Training Gopher.” *arXiv preprint*

Link: <https://arxiv.org/abs/2112.11446>

[Hoffmann et al., 2022] “Training Compute-Optimal Large Language Models.” *arXiv preprint*

Link: <https://arxiv.org/abs/2203.15556>

[Chowdhery et al., 2022] “PaLM: Scaling Language Modeling with Pathways.” *arXiv preprint*

Link: <https://arxiv.org/abs/2204.02311>

[Barham et al., 2022] “PATHWAYS: Asynchronous Distributed Dataflow for ML,” *MLSys 2022*

Link: <https://arxiv.org/abs/2203.12533>

[Touvron et al., 2023] “LLaMA: Open and Efficient Foundation Language Models.” *arXiv preprint*

Link: <https://arxiv.org/abs/2302.13971>

[Touvron et al., 2023] “Llama 2: Open Foundation and Fine-Tuned Chat Models.” *arXiv preprint*

Link: <https://arxiv.org/abs/2307.09288>

[Zhao et al., 2021] “Calibrate Before Use: Improving Few-Shot Performance of Language Models.” *ICML 2021*

Link: <https://arxiv.org/pdf/2102.09690>

[Ouyang et al., 2022] “Training Language Models to Follow Instructions with Human Feedback” *NeurIPS 2022*

Link: <https://arxiv.org/abs/2203.02155>

## References

---

[Wei et al., 2022] “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models” *NeurIPS 2022*

Link: <https://arxiv.org/abs/2201.11903>

[Wang et al., 2023] “Self-Consistency Improves Chain of Thought Reasoning in Language Models” *ICLR 2023*

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

[Kozima et al., 2022] “Large Language Models are Zero-Shot Reasoners” *NeurIPS 2022*

Link: <https://arxiv.org/abs/2205.11916>

[Wei et al., 2022] “Finetuned Language Models Are Zero-Shot Learners” *ICLR 2022*

Link: <https://arxiv.org/abs/2109.01652>

[Chung et al., 2022] “Scaling Instruction-Finetuned Language Models” *arXiv preprint*

Link: <https://arxiv.org/abs/2210.11416>

[Zhang et al., 2022] “DialogPT: Large-Scale Generative Pre-training for Conversational Response Generation” *ACL 2020*

Link: <https://arxiv.org/abs/1911.00536>

[Shuster et al., 2022] “BlenderBot 3: a deployed conversational agent that continually learns to responsibly engage” *arXiv preprint*

Link: <https://arxiv.org/abs/2208.03188>

[Thoppilan et al., 2022] “LaMDA: Language Models for Dialog Applications” *arXiv preprint*

Link: <https://arxiv.org/abs/2201.08239>

[Guu et al., 2020] “REALM: Retrieval-Augmented Language Model Pre-Training” *ICML 2020*

Link: <https://arxiv.org/pdf/2002.08909>

[Izacard et al., 2022] “Atlas: Few-shot Learning with Retrieval Augmented Language Models” *TMLR 2022*

Link: <https://arxiv.org/pdf/2208.03299>



## References

---

[Izacard et al., 2021] “Unsupervised Dense Information Retrieval with Contrastive Learning” *TMLR*

Link: <https://arxiv.org/abs/2112.09118>

[He et al., 2020] “Momentum Contrast for Unsupervised Visual Representation Learning” *CVPR 2020*

Link: <https://arxiv.org/abs/1911.05722>

[Izacard et al., 2020] “Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering” *EACL 2021*

Link: <https://arxiv.org/pdf/2007.01282>

[Raffel et al., 2019] “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer” *JMLR*

Link: <https://arxiv.org/abs/1910.10683>

[Lazaridou et al., 2022] “Internet-augmented Language Models through Few-shot Prompting for Open-domain Question Answering” *Deepmind*

Link: <https://arxiv.org/pdf/2203.05115>

[Shi et al., 2023] “REPLUG: Retrieval-Augmented Black-Box Language Models” *EMNLP 2023 Findings*

Link: <https://arxiv.org/pdf/2301.12652>

[Gao et al., 2023] “PAL: Program-aided Language Models.” *ICML 2023*

Link: <https://arxiv.org/abs/2211.10435>

[Madaan et al., 2023] “SELF-REFINE: Iterative Refinement with Self-Feedback.” *NeurIPS 2023*

Link: <https://arxiv.org/abs/2303.17651>

[Zhang et al., 2024] “Accessing GPT-4 level Mathematical Olympiad Solutions via Monte Carlo Tree Self-refine with LLaMa-3 8B: A Technical Report”

Link: <https://arxiv.org/abs/2406.07394>

[Yang et al., 2024] “Large Language Models as Optimizers.” *ICLR 2024*

Link: <https://arxiv.org/abs/2309.03409>

## References

---

[Park et al., 2023] “Generative Agents: Interactive Simulacra of Human Behavior.” *UIST 2023*

Link: <https://arxiv.org/abs/2304.03442>

[Huang et al., 2023] “Benchmarking Large Language Models As AI Research Agents.” *ICLR 2024 Submitted*

Link: <https://arxiv.org/abs/2310.03302>

[Yao, Shunyu, et al] “Tree of thoughts: Deliberate problem solving with large language models.” *Neurips 2024*

Link: <https://arxiv.org/abs/2305.10601>

[Rafailov, Rafael, et al] “Direct preference optimization: Your language model is secretly a reward model.” *Neurips 2024*

Link: <https://arxiv.org/abs/2305.18290>

[Dubey, et al] “The Llama 3 Herd of Models” *Meta Team Technical Report*

Link: <https://arxiv.org/abs/2407.21783>