

# 大規模言語モデルにおける プロンプトエンジニアリングの概説

東京大学大学院工学系研究科

技術経営戦略学専攻

松尾研究室 特任研究員

小島 武

# 概要

- プロンプトエンジニアリングとは
- 特徴（従来の最適化手法との比較）
- なぜ有効なのか？
- 大規模言語モデルにおける事例集
- 限界と可能性

# プロンプトエンジニアリングとは

- モデルから所望の出力を得るために良い入力文(プロンプト)を探索する作業.

User help me write a short note to introduce myself to my neighbor

ChatGPT Hi there,

My name is [Your Name] and I just moved in next door. I wanted to reach out and introduce myself. I'm looking forward to getting to know you and the neighborhood. If you ever need anything or just want to chat, please don't hesitate to stop by or give me a call.

Best,  
[Your Name]

User can you make it more formal?

ChatGPT Dear Neighbor,

I am writing to introduce myself. My name is [Your Name] and I have recently moved in next door. I wanted to take a moment to reach out and say hello. I am looking forward to getting to know you and the neighborhood.

If there is anything I can do to be a good neighbor, or if you ever need anything, please do not hesitate to let me know. I am happy to help in any way that I can.

An astronaut riding a horse in photorealistic style.



# プロンプトエンジニアリングとは

- Zero-Shot (Prompting) と Few-shot (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
```

## 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
```

# 特徴（従来の最適化手法との比較）

	プロンプトエンジニアリング	深層学習における最適化（確率的勾配法）
モデルのパラメータ更新	 不要 <ul style="list-style-type: none"><li>- 計算負荷からの解放&amp;パラメータ管理不要</li></ul>	 必要
学習サンプルサイズ	 zero shot~few shot <ul style="list-style-type: none"><li>- 様々なタスクに直面したときに有効</li><li>- サンプルが集めにくいタスクに有効（e.g.思考の連鎖）</li></ul>	 few shot~many shot
解釈性	 高い <ul style="list-style-type: none"><li>- 試行錯誤しやすい</li><li>- “モデルではなく人間がFineTuningされている”</li></ul>	 低い <ul style="list-style-type: none"><li>- ハイパラ</li></ul>

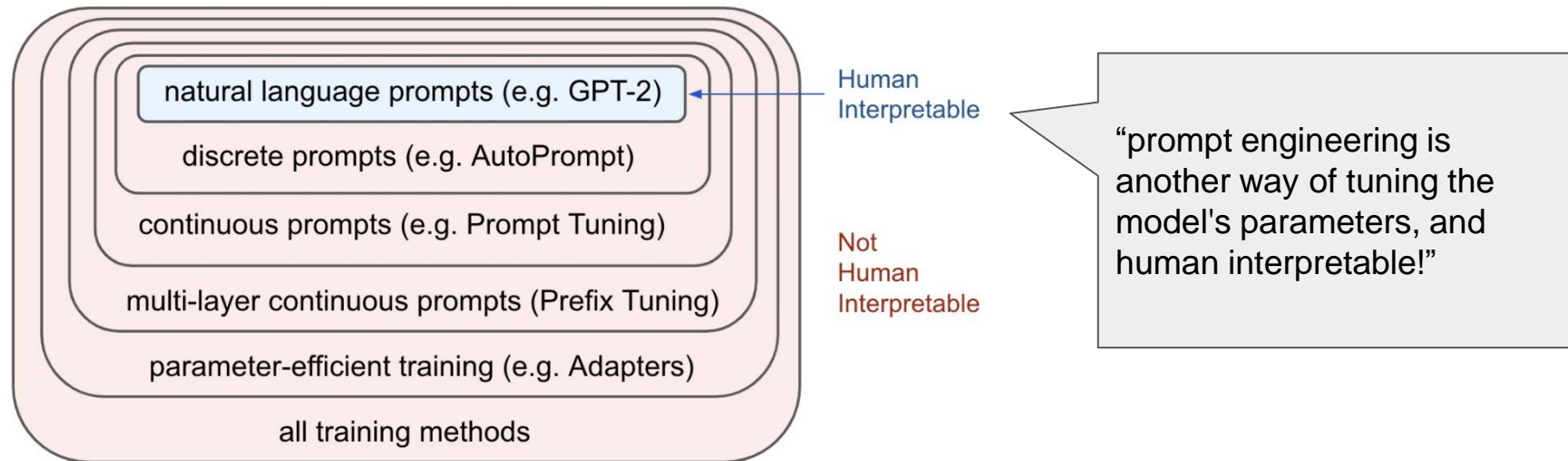
\*後ほど、プロンプトエンジニアの短所について「限界と可能性」の頁で言及します。

# 特徴（従来の最適化手法との比較）

## A Taxonomy of Prompting Methods

By Graham Neubig (10/15/2022)

See [CMU ANLP Prompting Lecture](#), [A Unified View of Parameter-Efficient Transfer Learning](#)



GPT-2: <https://openai.com/blog/better-language-models/>

AutoPrompt: <https://arxiv.org/abs/2010.15980>

Prefix Tuning: <https://arxiv.org/abs/2101.00190>

Prompt Tuning: <https://arxiv.org/abs/2104.08691>

Adapters: <https://arxiv.org/abs/2010.15980>

<https://twitter.com/gneubig/status/1581976078519742464>

# なぜ有効なのか？

- 大規模言語モデルの事前学習にて、大規模データで次の単語をひらすら予測する学習を行う → Zero-shot, Few-shotの能力が発現する (\*1).

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

(\*1) Language Models are Unsupervised Multitask Learners [\[URL\]](#)  
Language Models are Few-Shot Learners [\[URL\]](#)

## ➤ なぜ発現するのか？

- Zero-shot : 直感的に受け入れられる.
  - (大規模データから必要な情報を記憶し、表現を汎化してるのだろう)
- Few-shot : 直感的には理解しがたい.
  - 情報理論による分析 (\*2) : "言語のように学習対象が構成性を持つ場合、条件部 (=プロンプト) の後続 (=回答) の予測誤差は条件部を生成する構造の複雑さで抑えられることから説明できる" \* [\[Twitter\]](#)から文引用.

(\*2) A Theory of Emergent In-Context Learning as Implicit Structure Induction [\[URL\]](#)

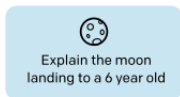
# なぜ有効なのか？

- 大規模言語モデルの事後学習にて, 人間の指示(質問)に従うようチューニング.

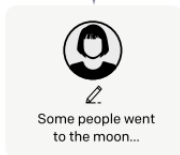
Step 1

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

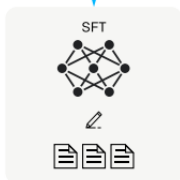
A prompt is  
sampled from our  
prompt dataset.



A labeler  
demonstrates the  
desired output  
behavior.



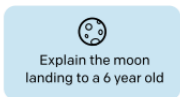
This data is used  
to fine-tune GPT-3  
with supervised  
learning.



Step 2

**Collect comparison data,  
and train a reward model.**

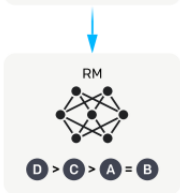
A prompt and  
several model  
outputs are  
sampled.



A labeler ranks  
the outputs from  
best to worst.



This data is used  
to train our  
reward model.



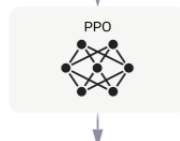
Step 3

**Optimize a policy against  
the reward model using  
reinforcement learning.**

A new prompt  
is sampled from  
the dataset.

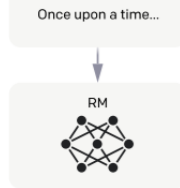


The policy  
generates  
an output.

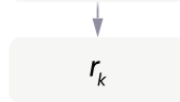


Once upon a time...

The reward model  
calculates a  
reward for  
the output.



The reward is  
used to update  
the policy  
using PPO.



Zero-shot, Few-shotの  
能力が改善.

Training language models to follow  
instruction with human feedback  
<https://arxiv.org/abs/2203.02155>



# 大規模言語モデルにおける事例集

- 思考の連鎖 (Chain-of-Thought: CoT) @Few-shot

## Standard Prompting

### Model Input

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

A: The answer is 11.

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

### Model Output

A: The answer is 27. ❌

## Chain-of-Thought Prompting

### Model Input

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

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

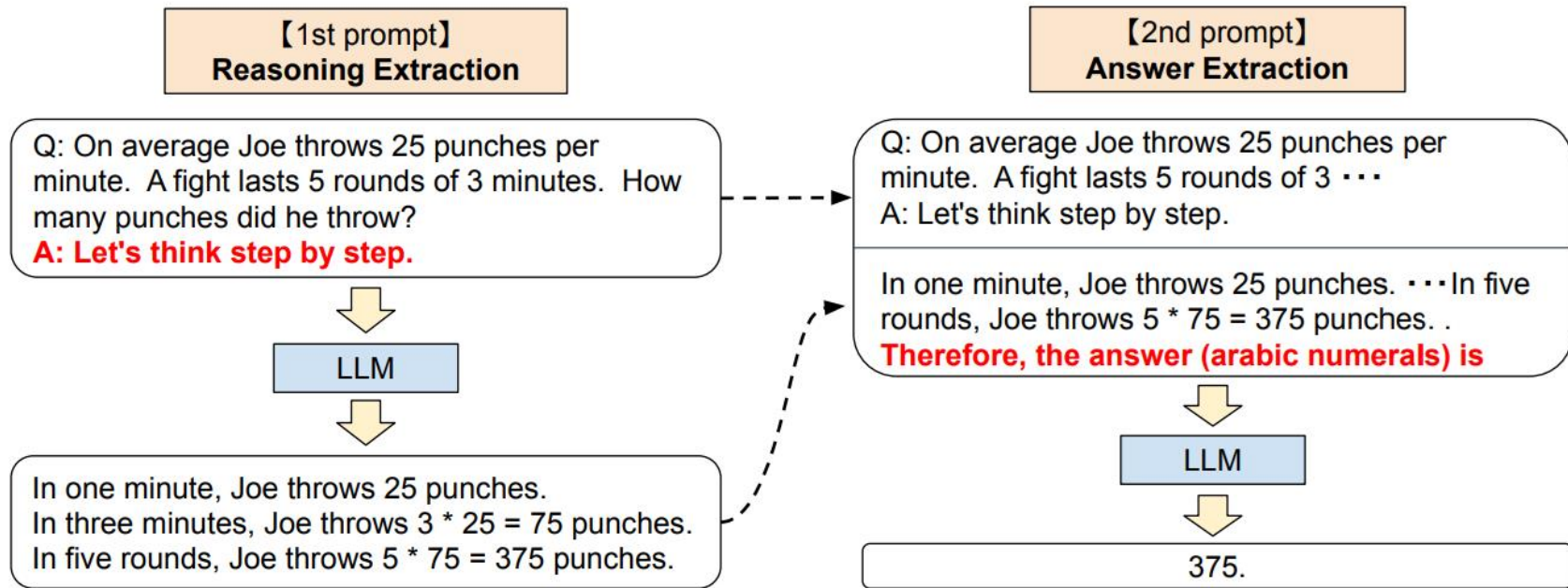
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

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

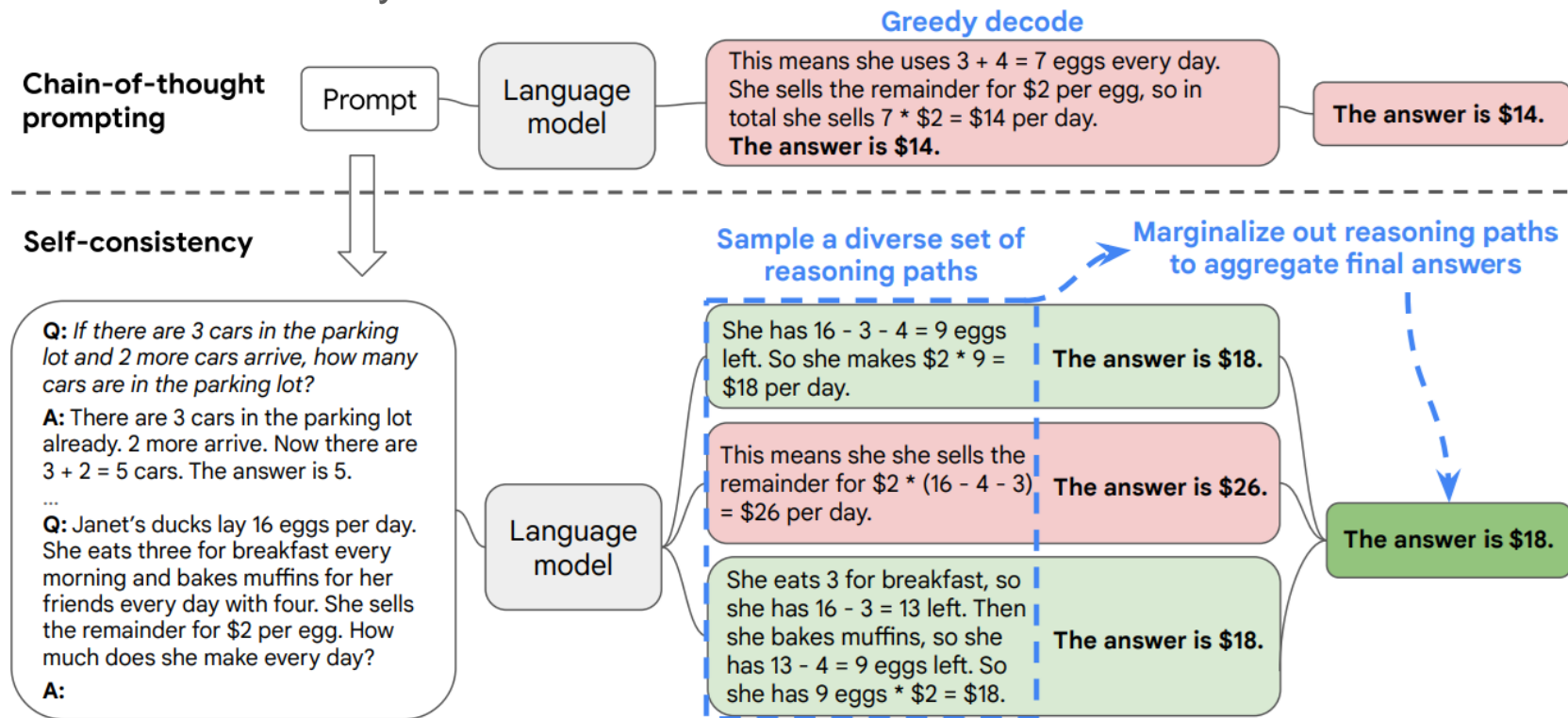
# 大規模言語モデルにおける事例集

- 思考の連鎖 (Chain-of-Thought: CoT) @Zero-shot



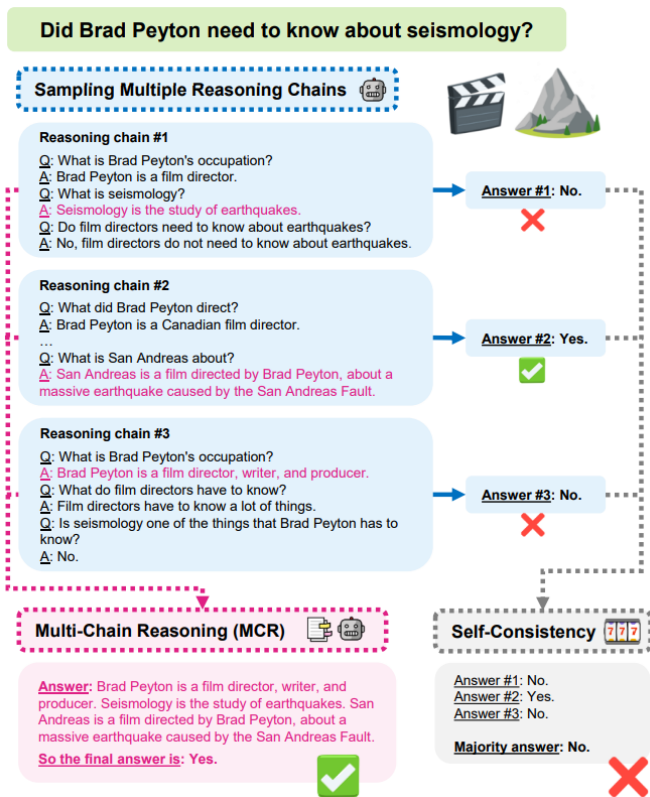
# 大規模言語モデルにおける事例集

- Self-Consistency : ランダムに思考の連鎖を複数回出力して多数決.



# 大規模言語モデルにおける事例集

- 多数決（Self-Consistency）の代わりに大規模言語モデルが最終判断.



Answering Questions by Meta-Reasoning over Multiple Chains of Thought

# 大規模言語モデルにおける事例集

- Program of Thought (PoT) : 言語モデルがプログラムを書く。

Question: In Fibonacci sequence, it follows the rule that each number is equal to the sum of the preceding two numbers. Assuming the first two numbers are 0 and 1, what is the 50th number in Fibonacci sequence?

The first number is 0, the second number is 1, therefore, the third number is  $0+1=1$ . The fourth number is  $1+1=2$ . The fifth number is  $1+2=3$ . The sixth number is  $2+3=5$ . The seventh number is  $3+5=8$ . The eighth number is  $5+8=13$ .  
..... (Skip 1000 tokens)  
The 50th number is 32,432,268,459.

CoT

↓  
32,432,268,459 ❌

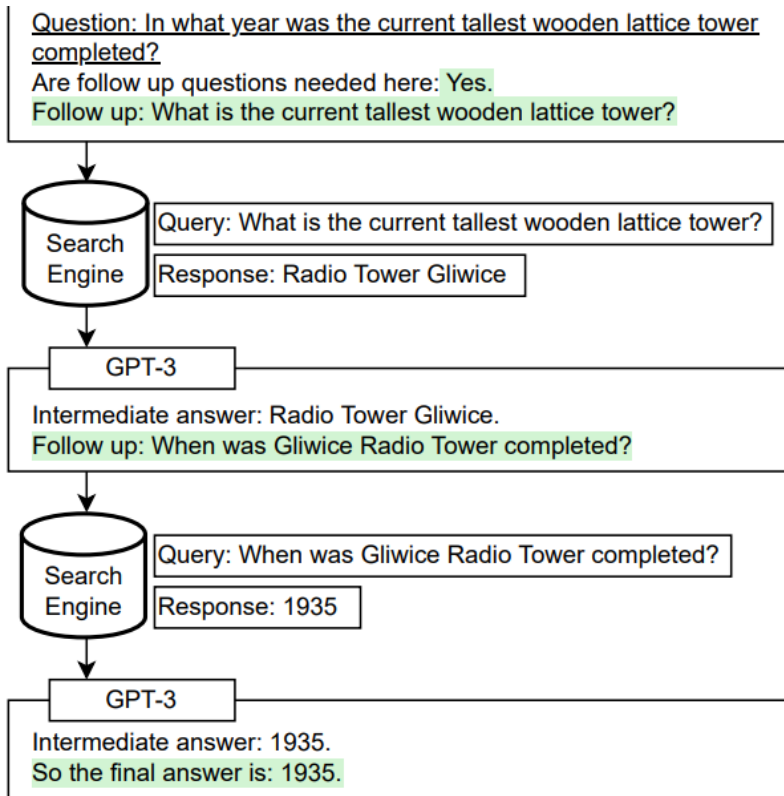
```
length_of_fibonacci_sequence = 50
fibonacci_sequence = np.zeros(length_of_)
fibonacci_sequence[0] = 0
fibonacci_sequence[1] = 1
For i in range(3, length_of_fibonacci_sequence):
    fibonacci_sequence[i] = fibonacci_sequence[i-1] +
    fibonacci_sequence[i-2]
ans = fibonacci_sequence[-1]
```

PoT

python ↓  
12,586,269,025 ✅

# 大規模言語モデルにおける事例集

- 知識を外部から借用しながら（検索しながら），“思考の連鎖”を行う。



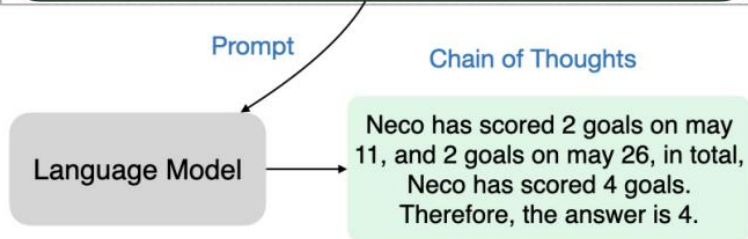
# 大規模言語モデルにおける事例集

- 表データを（一行ずつ）読み込んで思考の連鎖を行う。

Test Case				
date	result	score	brazil scorers	competition
may 11 , 1919	w	6 - 0	friedenreich (3) , neco (2) , harold	american championship
may 18 , 1919	w	3 - 1	heitor , amílcar , millon	american championship
may 26 , 1919	d	2 - 2	neco (2)	american championship
may 29 , 1919	w	1 - 0	friedenreich	american championship
june 1 , 1919	d	3 - 3	haroldo , arlindo (2)	taça roberto cherry

Brazilian football in 1919

How many goals have has Brazillian team player neco scored in 1919 south american championship?



# 大規模言語モデルにおける事例集

## ● 文章と画像を読み込んで、思考の連鎖を行う (GPT-4)

Answer question I.1.a. Think step-by-step.

### I. Principe de la détection de rayonnement avec un bolomètre

Comme illustré sur la figure 1 un bolomètre est constitué d'un absorbeur qui reçoit le rayonnement que l'on désire détecter. Sa température  $T$ , supposée uniforme, est mesurée à l'aide d'un thermomètre incorporé, constitué d'un matériau conducteur dont la résistance  $R(T)$  varie avec la température  $T$ ; cette variation est caractérisée par le coefficient  $\alpha = \frac{1}{R} \frac{dR}{dT}$ . L'ensemble possède la capacité thermique  $C_{th}$ .

Un barreau, conducteur thermique, homogène, de longueur  $L$ , de section  $S$  et de conductivité thermique  $\lambda$  et sans échanges thermiques latéraux, relie le bolomètre à un thermostat de température  $T_0$  fixe.

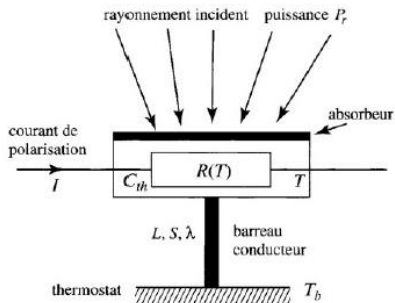


Figure 1 - Schéma d'un bolomètre

I.1. Dans cette première partie, on étudie le comportement du bolomètre en régime stationnaire, à la température de fonctionnement  $T = T_0$ .

I.1.a) Exprimer la température  $T(x)$  en tout point du barreau conducteur repéré par son abscisse  $x$ ; on prendra  $T(0) = T_0$  et  $T(L) = T_b$ .

To answer question I.1.a, we need to express the temperature  $T(x)$  in every point of the conductive bar, which is characterized by the abscissa  $x$ . We are given that  $T(0) = T_0$  and  $T(L) = T_b$ .

For a stationary regime, the heat equation for the one-dimensional heat conduction in the bar is:

$$d^2T/dx^2 = 0$$

Integrating once, we get:

$$dT/dx = C_1$$

Integrating a second time:

$$T(x) = C_1x + C_2$$

To determine the constants  $C_1$  and  $C_2$ , we use the boundary conditions:

$$T(0) = T_0, \text{ which gives } C_2 = T_0$$

$$T(L) = T_b, \text{ which gives } C_1 = (T_b - T_0)/L$$

Finally, the expression for  $T(x)$  is:

$$T(x) = (T_b - T_0) * (x/L) + T_0$$

This represents a linear temperature profile along the conductive bar, with a slope of  $(T_b - T_0)/L$ .



# 限界と可能性

- 長文対応

- コンテキスト(入力+出力)長に限界がある. [\[URL\]](#)

LATEST MODEL	DESCRIPTION	MAX TOKENS	TRAINING DATA
gpt-4	More capable than any GPT-3.5 model, able to do more complex tasks, and optimized for chat. Will be updated with our latest model iteration.	8,192 tokens	Up to Sep 2021
gpt-4-0314	Snapshot of gpt-4 from March 14th 2023. Unlike gpt-4, this model will not receive updates, and will be deprecated 3 months after a new version is released.	8,192 tokens	Up to Sep 2021
gpt-4-32k	Same capabilities as the base gpt-4 mode but with 4x the context length. Will be updated with our latest model iteration.	32,768 tokens	Up to Sep 2021
gpt-4-32k-0314	Snapshot of gpt-4-32k from March 14th 2023. Unlike gpt-4-32k, this model will not receive updates, and will be deprecated 3 months after a new version is released.	32,768 tokens	Up to Sep 2021

# 限界と可能性

---

## Scaling Transformer to 1M tokens and beyond with RMT

---

- 長文対応
  - 限界を突破する研究も出始めている。[URL]

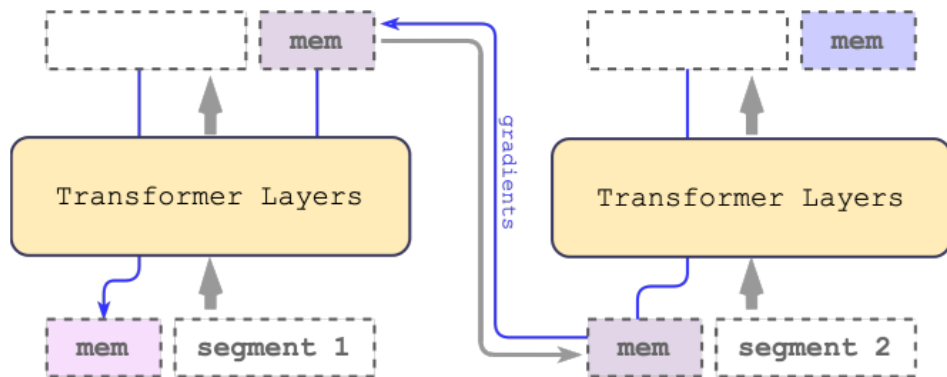
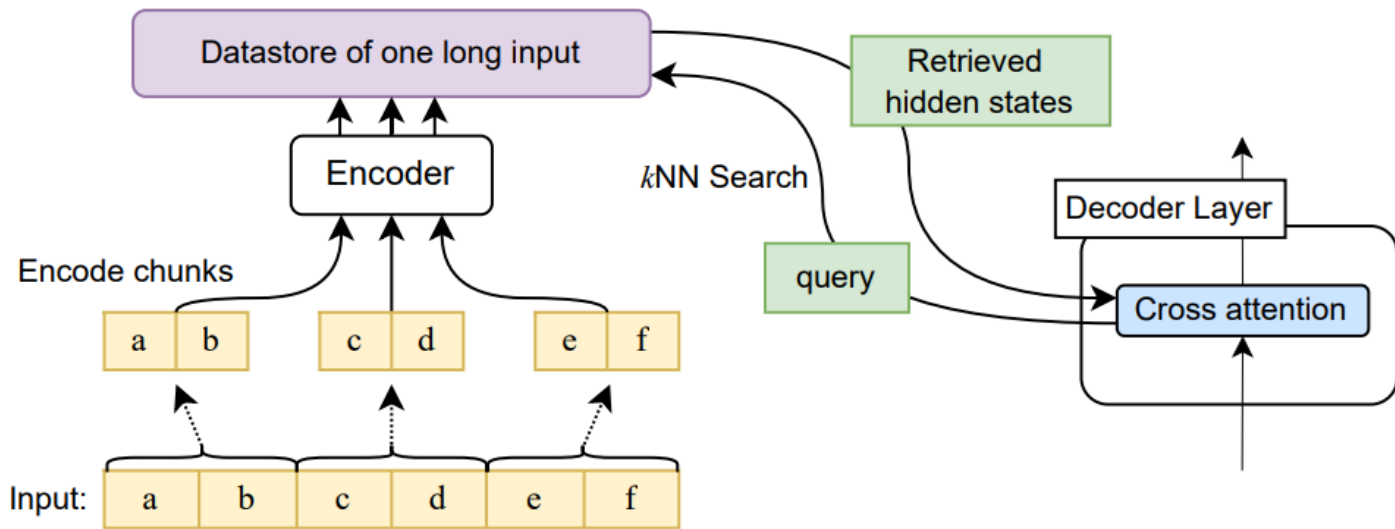


Figure 2: **Recurrent memory mechanism.** Memory is passed to Transformer along input sequence embeddings, and memory output is passed to the next segment. During training gradients flow from the current segment through memory to the previous segment.

# 限界と可能性

## Unlimiformer: Long-Range Transformers with Unlimited Length Input

- 長文対応
  - 限界を突破する研究も出始めている。[URL]



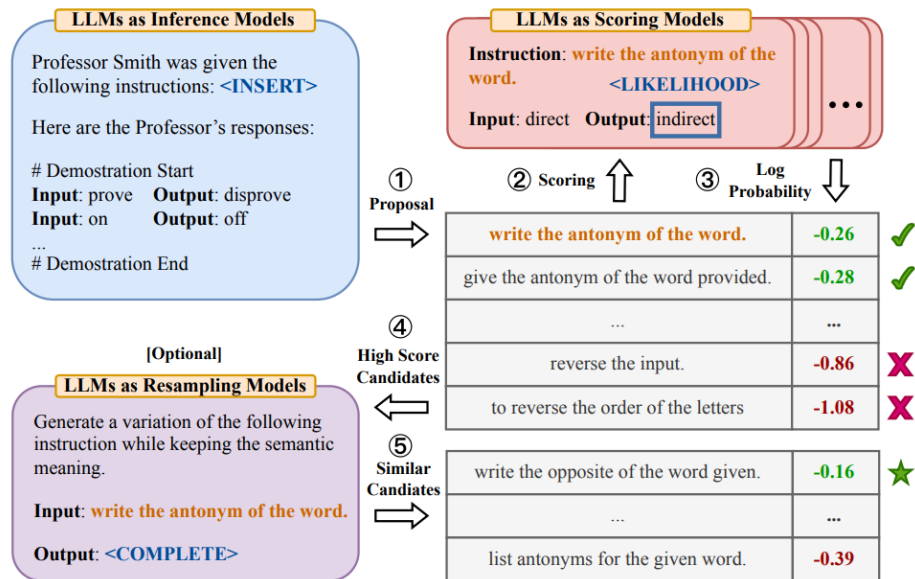
# 限界と可能性

## LARGE LANGUAGE MODELS ARE HUMAN-LEVEL PROMPT ENGINEERS

- 人手作業

- 良いプロンプトは基本的に人手で探す必要がある。
- 自動的により良いプロンプトを探す研究もある。[[URL](#)]

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



# 限界と可能性

- プロンプトの頑健性

- (Zero-shot) テンプレートに敏感. [\[URL\]](#) (Appendix A)
- (Few-shot) デモの順番に敏感. [\[URL\]](#)

	FLAN 137B													
	GLaM		LaMDA-PT		GPT-3 175B		zero-shot				few-shot			
	Random Guess	Supervised Model	zero-shot	one-shot	zero-shot	few-shot [k]	zero-shot	few-shot [k]	average template	best dev template	average template	best dev template	[k]	#t
<b>NLI</b>														
ANLI R1	33.3	57.4 <sup>b</sup>	40.9	42.4	39.6	39.0 [5]	34.6	36.8 [50]	47.7±1.4	46.4	44.2±2.3	47.9	[6]	8
ANLI R2	33.3	48.3 <sup>b</sup>	38.2	40.0	39.9	37.5 [5]	35.4	34.0 [50]	43.9±1.3	44.0	41.6±1.4	41.1	[6]	8
ANLI R3	33.3	43.5 <sup>b</sup>	40.9	40.8	39.3	40.7 [5]	34.5	40.2 [50]	47.0±1.3	48.5	42.8±2.2	46.8	[6]	8
CB	33.3	93.6 <sup>a</sup>	33.9	73.2	42.9	34.4 [5]	46.4	82.1 [32]	64.1±14.7	83.9	82.6±4.4	82.1	[7]	10
MNLI-m	33.3	92.2 <sup>a</sup>	-	-	35.7	43.7 [5]	-	-	51.1±6.2	61.2	60.8±3.7	63.5	[10]	10
MNLI-mm	33.3	91.9 <sup>a</sup>	-	-	37.0	43.8 [5]	-	-	51.0±6.5	62.4	61.0±3.5	63.5	[10]	10
QNLI	50.0	96.9 <sup>a</sup>	-	-	50.6	55.7 [5]	-	-	59.6±4.9	66.4	62.0±1.7	63.3	[12]	9
RTE	50.0	92.5 <sup>a</sup>	68.8	71.5	73.3	70.8 [5]	63.5	72.9 [32]	78.3±7.9	84.1	79.9±6.9	84.5	[8]	10
SNLI	33.3	91.3 <sup>b</sup>	-	-	33.3	54.7 [5]	-	-	43.0±7.4	53.4	62.3±2.4	65.6	[15]	9
WNLI	50.0	94.5 <sup>a</sup>	-	-	56.3	64.8 [5]	-	-	61.0±10.6	74.6	55.4±11.0	70.4	[14]	10
<b>READING COMP.</b>														
BoolQ	50.0	91.2 <sup>a</sup>	83.0	82.8	81.0	80.0 [1]	60.5	77.5 [32]	80.2±3.1	82.9	83.6±0.8	84.6	[4]	9
DROP	-	80.5 <sup>b</sup>	54.9	55.2	3.8	10.3 [1]	23.6 <sup>†</sup>	36.5 [20]	21.9±0.9	22.7	22.3±1.1	23.9	[2]	7
MultiRC	-	88.1 <sup>a</sup>	45.1	62.0	60.0	59.6 [5]	72.9	74.8 [32]	74.5±3.7	77.5	69.2±3.2	72.1	[1]	8

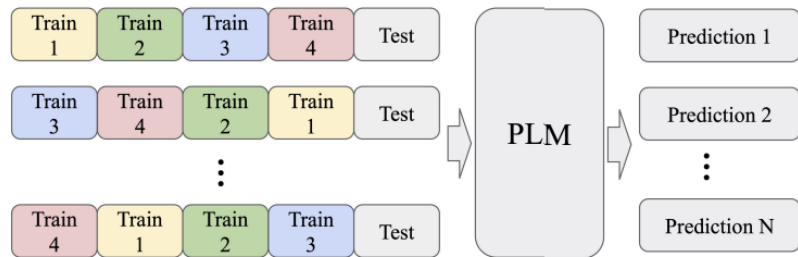


Figure 2: Training sample permutations for the In-context Learning setting. The concatenation of training samples as well as test data transforms the classification task into a sequence generation task.

# 限界と可能性

- マルチモダリティへの対応

- GPT4は画像と文章を同時に入力することができる。[URL](#)

---

**GPT-4 visual input example, Extreme Ironing:**

---

User      What is unusual about this image?



Source: <https://www.barnorama.com/wp-content/uploads/2016/12/03-Confusing-Pictures.jpg>

GPT-4      The unusual thing about this image is that a man is ironing clothes on an ironing board attached to the roof of a moving taxi.

---

**Table 16.** Example prompt demonstrating GPT-4's visual input capability. The prompt requires image understanding.

# まとめ

- プロンプトエンジニアリングとは、モデルから所望の出力を得るために最適な入力文を探索する作業
- プロンプトエンジニアリングは、パラメータを更新せずにモデルの挙動を変更できる点、限られたサンプル数（ゼロショットもしくはヒューショット）で有効である点、人間の解釈性が高いという点で、従来の深層学習の勾配法による最適化手法とは異なる特徴を持つ。
- 大規模言語モデルは段階的な思考を必要とするような複雑なタスクにおいて有効であることが実証されているが、大規模言語モデルにおけるプロンプトエンジニアリングが、そのような複雑なタスクを解くためにどのように利用されているか事例にて解説。
- 大規模言語モデルのプロンプトエンジニアについての今後の限界と可能性として、長文への対応、人手作業の必要、プロンプトの頑健性などが挙げられる。