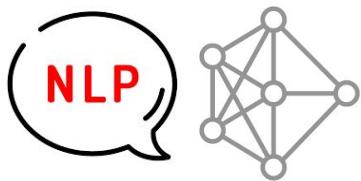




CHULA ENGINEERING
Foundation toward Innovation

COMPUTER



Prompting

2110572: Natural Language Processing Systems

Peerapon Vateekul & Ekapol Chuangsawanich

Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University



Outline

- Introduction
- In-context Learning (ICL)
- Prompt Engineering
- Advanced Prompting Techniques
- Evaluation

+

Introduction



Problems with current LLMs

- Language models learn from the data they are trained on.
- But our knowledge is constantly changing.
 - E.g., in 2006, Pluto was declared *not* a planet; however, if we trained a language model on data prior to 2006, this is what we would get when we asked it:





How to update LLM knowledge?

- Train from scratch again?
 - Way too expensive
- Continue training on new data?
 - LMs forget old data (catastrophic forgetting)
- In-context learning!

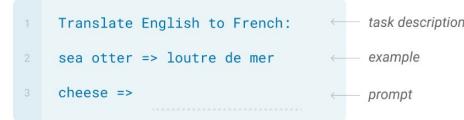
Zero-shot

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



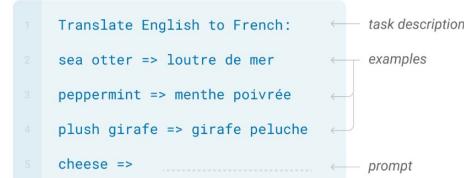
One-shot

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



Few-shot

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



+

In-context Learning (ICL)

Arxiv-2020

EMNLP-2022

ICLR-2023

EMNLP-2024

In-context learning (ICL): Example of ICL (GPT3, Arxiv-2020)

- ICL is an emergent ability of LLMs popularized in the GPT3 paper.
- LLMs can “learn” new tasks from prompts *without training*.

Language Models are Few-Shot Learners

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

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely

05.14165v4 [cs.CL] 22 Jul 2020

The three settings we explore for in-context learning

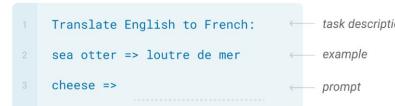
Zero-shot

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



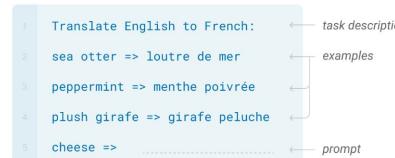
One-shot

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



Few-shot

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



Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.





In-context learning (ICL): EMNLP-2022

- ICL is hypothesized to be learned during the pretraining stage ([Min et al. \(2022, Best paper at EMNLP\)](#)[1]).
- But the LLM is not actually *learning*, i.e., their weights don't change.

**Rethinking the Role of Demonstrations:
What Makes In-Context Learning Work?**

Sewon Min^{1,2} Xinxi Lyu¹ Ari Holtzman¹ Mikel Artetxe²
Mike Lewis² Hannaneh Hajishirzi^{1,3} Luke Zettlemoyer^{1,2}

¹University of Washington ²Meta AI ³Allen Institute for AI
{sewon,alrope,ahai,hannaneh,lsz}@cs.washington.edu
{artetxe,mikelewis}@meta.com

Abstract

Large language models (LMs) are able to in-context learn—perform a new task via inference alone by conditioning on a few input-label pairs (demonstrations) and making predictions for new inputs. However, there has been little understanding of *how* the model learns and *which* aspects of the demonstrations contribute to end task performance. In this paper, we show that ground truth demonstrations are in fact not required—randomly replacing labels in the demonstrations barely hurts performance on a range of classification and multi-choice tasks, consistently over 12 different models including GPT-2. Instead, we

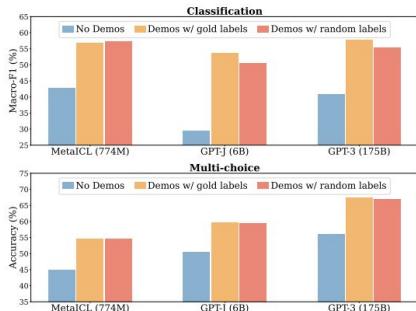


Figure 1: Results in classification (top) and multi-

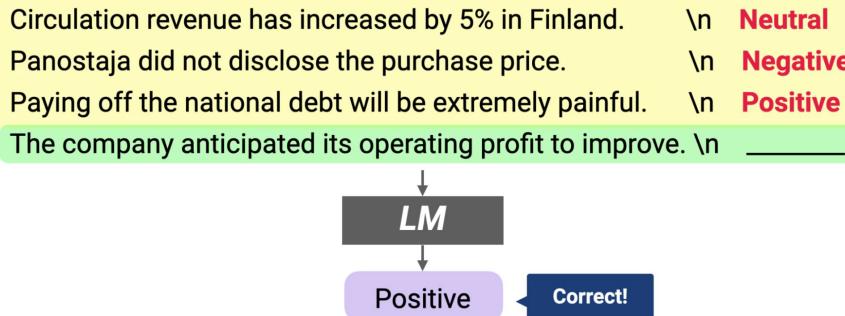
6 Discussion & Conclusion

In this paper, we study the role of the demonstrations with respect to the success of in-context learning. We find that the ground truth input-label mapping in the demonstrations matters significantly less than one might think—replacing gold labels with random labels in the demonstrations only marginally lowers the performance. We then identify a series of aspects in the demonstrations and examine which aspect actually contributes to performance gains. Results reveal that (1) gains are mainly coming from *independent* specification of the input space and the label space, (2) the models can still retain up to 95% of performance gains by using either the inputs only or the label set only if the right format is used, and (3) meta-training with an in-context learning objective magnifies these trends. Together, our findings lead to a set of broader indications about in-context learning, as well as avenues for future work.



In-context learning (ICL)

- There are many hypotheses on how ICL works:
- [1] showed that ICL acts more as a **pattern recognition procedure**.
 - **The input-output mappings** that are provided allow a model to retrieve similar examples it has been exposed to during training, but the moment you start flipping labels or the template model performance breaks.
- **Example 1: Maintain the output labels, but randomly assign the ground truth**
- **The result is still correct!**



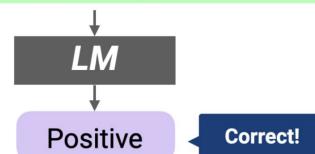


In-context learning (ICL)

- There are many hypotheses on how ICL works:
- [1] showed that ICL acts more as a **pattern recognition procedure**.
 - The **input-output mappings** that are provided allow a model to retrieve similar examples it has been exposed to during training, but the moment you start flipping labels or the template model performance breaks.
- **Example2: Completely random the output labels**
- **The result is now incorrect!**

Circulation revenue has increased by 5% in Finland. \n **Neutral**
 Panostaja did not disclose the purchase price. \n **Negative**
 Paying off the national debt will be extremely painful. \n **Positive**

The company anticipated its operating profit to improve. \n _____



Circulation revenue has increased by 5% in Finland. \n **Unanimity**
 Panostaja did not disclose the purchase price. \n **Wave**
 Paying off the national debt will be extremely painful. \n **Guana**

The company anticipated its operating profit to improve. \n _____



[1] <https://arxiv.org/abs/2202.12837>

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



In-context learning (ICL): ICLR-2023

- While [1] shows that ICL behaves similarly to fine-tuning.
 - ICL tends to change the attention distribution in the same way as fine-tuning during forward computation.

Published at the Workshop on Understanding Foundation Models at ICLR 2023

WHY CAN GPT LEARN IN-CONTEXT? LANGUAGE MODELS IMPLICITLY PERFORM GRADIENT DESCENT AS META-OPTIMIZERS

Damai Dai^{†*}, Yutao Sun^{||*}, Li Dong[‡], Yaru Hao[‡], Shuming Ma[‡], Zhifang Sui[†], Furu Wei[‡]

[†] Peking University ^{||} Tsinghua University [‡] Microsoft Research

{daidamai,szf}@pku.edu.cn

{lidong1,fuwei}@microsoft.com

<https://github.com/microsoft/LMOps>

ABSTRACT

[1] <https://openreview.net/forum?id=fzbHRjAd8U> Large pretrained language models have shown surprising in-context learning (ICL) ability. With a few demonstration input-label pairs, they can predict labels for many new inputs. This ability has been widely used in various applications.

+ In-context learning (ICL)

- While [1] shows that ICL behaves similarly to fine-tuning.
 - ICL tends to change the attention distribution in the same way as fine-tuning during forward computation.

ABSTRACT

Large pretrained language models have shown surprising in-context learning (ICL) ability. With a few demonstration input-label pairs, they can predict labels for unseen inputs without parameter updates. Despite the great success in performance, its working mechanism still remains an open question. In this paper, we explain language models as meta-optimizers and understand ICL as implicit finetuning. Theoretically, we figure out that Transformer attention has a dual form of gradient descent. On top of it, we understand ICL as follows: GPT first produces meta-gradients according to the demonstration examples, and then these meta-gradients are applied to the original GPT to build an ICL model. We compare the behaviors of ICL and explicit finetuning on real tasks to provide empirical evidence that supports our understanding. Experimental results show that in-context learning behaves similarly to explicit finetuning from multiple perspectives.

$$\begin{aligned}
 \tilde{f}_{\text{ICL}}(\mathbf{q}) &= W_{\text{ZSL}}\mathbf{q} + W_V X' (W_K X')^T \mathbf{q} = W_{\text{ZSL}}\mathbf{q} + \text{LinearAttn}(W_V X', W_K X', \mathbf{q}) \\
 &= W_{\text{ZSL}}\mathbf{q} + \sum_i W_V \mathbf{x}_i' ((W_K \mathbf{x}_i')^T \mathbf{q}) = W_{\text{ZSL}}\mathbf{q} + \sum_i ((W_V \mathbf{x}_i') \otimes (W_K \mathbf{x}_i)) \mathbf{q} \\
 &= W_{\text{ZSL}}\mathbf{q} + \Delta W_{\text{ICL}}\mathbf{q} = (W_{\text{ZSL}} + \Delta W_{\text{ICL}})\mathbf{q}.
 \end{aligned} \tag{3}$$

[1] <https://openreview.net/forum?id=fzbHRjAd8U>

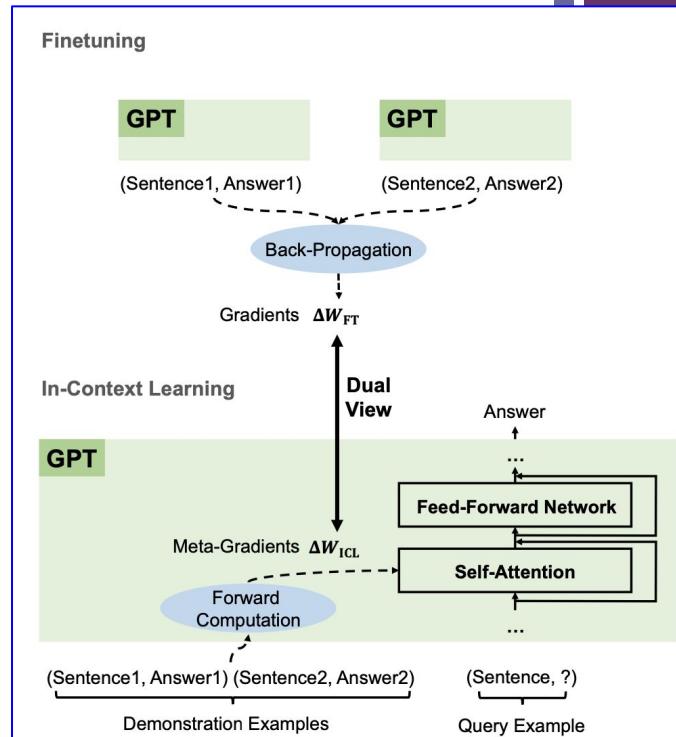


Figure 1: Through forward computation, GPT produces meta-gradients for ICL, which shares a dual view with fine-tuning that updates model parameters with back-propagated gradients.



In-context learning (ICL)

- While [1] shows that ICL behaves similarly to fine-tuning.
 - ICL tends to change the attention distribution in the same way as fine-tuning during forward computation.

Model	Metric	SST2	SST5	MR	Subj	AGNews	CB	Average
GPT 1.3B	SimAM (Before Finetuning)	0.555	0.391	0.398	0.378	0.152	0.152	0.338
	SimAM (After Finetuning)	0.585	0.404	0.498	0.490	0.496	0.177	0.442
GPT 2.7B	SimAM (Before Finetuning)	0.687	0.380	0.314	0.346	0.172	0.228	0.355
	SimAM (After Finetuning)	0.687	0.492	0.347	0.374	0.485	0.217	0.434

Table 3: SimAM on six datasets. From the perspective of attention behavior, ICL is more inclined to generate similar attention weights to those after finetuning.

SimAM = cosine similarity of the attention map **FT vs ICL**

+ In-context learning (ICL): EMNLP-2024

Still no consensus on how or why ICL works (EMNLP2024)

- Theoretical & empirical (experimental) perspectives.

The Mystery of In-Context Learning: A Comprehensive Survey on Interpretation and Analysis

Yuxiang Zhou[♡], Jiazheng Li[♡], Yanzheng Xiang[♡], Hanqi Yan[♡], Lin Gui[♡], Yulan He^{♡◊}

[♡]Department of Informatics, King's College London [◊]The Alan Turing Institute

{yuxiang.zhou,jiazheng.li,yanzheng.xiang}@kcl.ac.uk

{hanqi.yan,lin.1.gui,yulan.he}@kcl.ac.uk

Abstract

Understanding in-context learning (ICL) capability that enables large language models (LLMs) to excel in proficiency through demonstration examples is of utmost importance. This importance stems not only from the better utilization of this capability across various tasks, but also from the proactive identification and mitigation of potential risks, including concerns regarding truthfulness, bias, and toxicity, that may arise alongside the capability. In this paper, we present a thorough survey on the interpretation and analysis of in-context learning. First, we provide a concise introduction

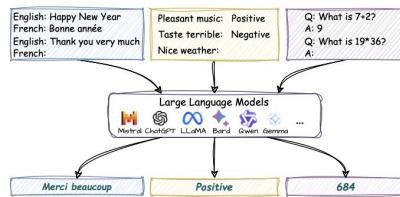


Figure 1: Illustration of In-context Learning.

Qwen2 (QwenTeam, 2024) have exhibited proficiency across various tasks, such as machine translation, sentiment analysis, and question answering,

Work	Key Words	Models	Tasks
Theoretical Perspective			
(Elhage et al., 2021)	Mechanistic Interpretability	Transformer [†]	-
(Olsson et al., 2022)	Mechanistic Interpretability	Transformer	-
(Edelman et al., 2024)	Mechanistic Interpretability	Transformer [†]	Markov Chain modeling
(Swaminathan et al., 2023)	Mechanistic Interpretability	Transformer	Next token prediction
(Todd et al., 2024)	Mechanistic Interpretability	Llama 2, GPT [†]	Antonym, etc.
(Bai et al., 2023)	Mechanistic Interpretability	Transformer [†]	Regression
(Garg et al., 2022)	Regression Function Learning	Transformer [†]	Regression
(Li et al., 2023b)	Regression Function Learning	Transformer [†]	Regression
(Li et al., 2023a)	Regression Function Learning	Transformer [†]	Regression
(Akkyurek et al., 2023)	Regression Function Learning	Transformer [†]	Regression
(Guo et al., 2024)	Gradient Descent, Meta-Optimization	GPT [†]	Representation Regression
(Dai et al., 2023)	Gradient Descent, Meta-Optimization	Transformer [†]	Classification
(von Oswald et al., 2023a)	Gradient Descent, Meta-Optimization	Transformer [†]	Regression
(von Oswald et al., 2023b)	Gradient Descent, Meta-Optimization	Transformer [†]	Regression
(Deutch et al., 2024)	Gradient Descent, Meta-Optimization	Transformer [†]	Classification
(Shen et al., 2024)	Gradient Descent, Meta-Optimization	LLM4, GPT [†]	Classification
(Fu et al., 2024)	Gradient Descent, Meta-Optimization	Transformer, LSTM	Regression
(Xie et al., 2022)	Bayesian Inference	Transformer [†] , LSTM	Next token prediction
(Wang et al., 2023b)	Bayesian Inference	GPT [†]	Classification
(Wies et al., 2023)	Bayesian Inference	-	-
(Jiang, 2023)	Bayesian Inference	GPT [†]	Synthetic Generation
(Zhang et al., 2023b)	Bayesian Inference	Transformer [†]	-
(Panwar et al., 2024)	Bayesian Inference	Transformer	Regression
(Jeon et al., 2024)	Bayesian Inference	Transformer [†]	Regression
(Bigelow et al., 2024)	Bayesian Inference	GPT [†]	Sequence generation
Empirical Perspective			
(Shin et al., 2022)	DATA Domain	GPT-3	Classification, etc.
(Han et al., 2023)	DATA Domain, DATA Distribution	GPT [†]	Classification
(Raventos et al., 2023)	Task Diversity	GPT-2	Regression
(Razeghi et al., 2022)	DATA Term frequency	GPT [†]	Reasoning
(Kandpal et al., 2023)	DATA Term frequency	GPT-3 [†]	Question Answering
(Chan et al., 2022)	DATA Distribution	Transformer	Classification
(Yadlowsky et al., 2023)	DATA Distribution	GPT [†]	Regression
(Hendel et al., 2023)	Task Diversity	LLAMA	Translation, etc.
(Wei et al., 2022b)	Model Scale	GPT ³	Classification
(Schaeffer et al., 2023)	Model Scale, Evaluation Metric	GPT-3 [†]	Classification
(Tay et al., 2023)	Pre-training Objective	UL2 [†]	Classification, etc.
(Kirsch et al., 2024)	Model Architecture	Transformer [†]	Classification
(Singh et al., 2023)	Model Architecture	LLMA [†]	Synthetic generation
(Yousefi et al., 2024)	Embeddings	Llama, Vicuna	Regression
(Akkyurek et al., 2024)	Model Architecture	Transformer, LSTM, etc.	Language learning
(Lu et al., 2022)	Demonstration Order	GPT [†]	Classification
(Liu et al., 2024)	Demonstration Order	GPT-3.5 [†]	Question Answering
(Zhao et al., 2021)	Demonstration	GPT [†]	Classification, IE, IR
(Liu et al., 2022)	Demonstration Order	GPT-3	Classification, QA, etc.
(Min et al., 2022)	Input-Label Mapping	GPT [†]	Classification
(Yoo et al., 2022)	Input-Label Mapping	GPT [†]	Classification
(Wei et al., 2023)	Input-Label Mapping	GPT-3 [†]	Classification, QA, etc.
(Pan et al., 2023)	Input-Label Mapping	GPT-3 [†]	Classification, etc.
(Lin and Lee, 2024)	Input-Label Mapping	LLama, LSTM, etc.	Classification, etc.
(Kossow et al., 2024)	Input-Label Mapping	LLMA [†]	Classification
(Tang et al., 2023)	Input-Label Mapping	GPT [†]	Classification
(Si et al., 2023)	Input-Label Mapping	GPT-3 [†]	Classification, QA, etc.
(Wang et al., 2023a)	Input-Label Mapping	GPT2-XL	Classification



In-context learning (ICL): Conclusion

- ICL allows us to use LLMs on any natural language task without having to fine-tune the models.
- Instead, we can only rely on supplying LLMs informative prompts.
 - The process of designing prompts is called **prompt engineering**.



Quick Guide: AI Prompt Engineering

Strategies for better results in ChatGPT, Perplexity and other AI tools.

EXAMPLE

You are an expert, creative writer, working on an **essay about pollution**. Create a table with 3 columns labelled nouns, verbs, adjectives. **Create a vocabulary list of 20 pollution words** for each column as a word cache for the essay . The essay is **showing the damage to habitat and animals**. It is **factual**, but will create a **sense of crisis**. The vocabulary list should **support those goals**.

Your Prompt should include

WHO: 
Role: Who is the tool writing as?
Audience: Who is it writing FOR?

WHAT: 
Task: What do you want the tool to do?
Voice: Should the tone be informal, chatty, professional?

WHY: 
What is the goal of the task?

WHERE: 
What is the context of the task?

PRO TIP
Use adjectives to help define your "role"

More details = better results. This takes some planning. Think the ideal response, what you want it to include, how you want it to look, how it should sound, and include that in the instructions.

Explaining the **purpose of the task** and what you're **trying to achieve** helps the AI Tool shape its response to better meet your needs.

Include necessary **background**, any **limitations** or things NOT to include. Again, this helps generate better results to fit the task.

To improve your results...

Include examples : show the AI what you want the response to look like/include.

Prompt Stacking: Too many directions can confuse the AI. Break the prompt into separate parts. The AI can use results from previous prompts to generate new tasks.

Explain Reasoning: Tell it to explain its thinking step-by-step.

Ask for multiple versions: Lets you pick and choose and build on the better options.

SOURCES: [UX Planet: Ultiage Guide](#); [AI Prompt Writing Formula](#)

©creative commons Jeri Hurd ,Branksome Hall Asia

+

Prompt Engineering



Prompt types

- 1) **User prompt:** the specific instructions or questions a user provides to an AI system to elicit a desired response.
- 2) **System prompt:** the foundational instructions that dictate an AI's behavior.

```

1 const response = await openai.chat.completions.create({
2   model: "gpt-4o",
3   messages: [
4     {
5       "role": "developer",
6       "content": [
7         {
8           "type": "text",
9           "text": `You are a helpful assistant that answers programming
10             questions in the style of a southern belle from the
11               southeast United States.
12             `,
13         }
14       ]
15     },
16   ],
17   "role": "user",
18   "content": [
19     {
20       "type": "text",
21       "text": "Are semicolons optional in JavaScript?"
22     }
23   ]
24 },
25 ],
26 store: true,
27 });
28 });

```

```

import anthropic

client = anthropic.Anthropic()

response = client.messages.create(
    model="claude-3-5-sonnet-20241022",
    max_tokens=2048,
    system="You are a seasoned data scientist at a Fortune 500 company.", # <-- role
    messages=[
        {"role": "user", "content": "Analyze this dataset for anomalies: <dataset>{"
    ]
)

print(response.content)

```

Elements of a Prompt

Instruction - a specific task or instruction you want the model to perform

Context - external information or additional context that can steer the model to better responses

Input Data - the input or question that we are interested to find a response for

Output Indicator - the type or format of the output.



Example 1

Instruction

Context

Input Data

Output Indicator

Classify the following text into
neutral, negative, or positive.

Text: I love this food.

Sentiment:



Example 2

Instruction

Answer the question using information from the given context.

Context

I love this food so much. Everything is great ...

Input Data

Question: Did the customer love the food?

Output Indicator

Answer:

Prompting principles

Principle 1: Write clear and specific instructions

Principle 2: Give the model time to “think”

Tactic 1.1: Delimiters to indicate parts of the input

Task: Summarize text

22

- Delimiters can be anything like: ```, """", <>, <tag> </tag>, :

```
In [3]: text = f"""
You should express what you want a model to do by \
providing instructions that are as clear and \
specific as you can possibly make them. \
This will guide the model towards the desired output, \
and reduce the chances of receiving irrelevant \
or incorrect responses. Don't confuse writing a \
clear prompt with writing a short prompt. \
In many cases, longer prompts provide more clarity \
and context for the model, which can lead to \
more detailed and relevant outputs.
"""

prompt = f"""
Summarize the text delimited by triple backticks \
into a single sentence.
``{text}``
"""

response = get_completion(prompt)
print(response)
```

You should express what you want a model to do by \ providing instructions that are as clear and \ specific as you can possibly make them. \ This will guide the model towards the desired output, \ and reduce the chances of receiving irrelevant \ or incorrect responses. Don't confuse writing a \ clear prompt with writing a short prompt. \ In many cases, longer prompts provide more clarity \ and context for the model, which can lead to \ more detailed and relevant outputs.

Summarize the text delimited by triple backticks \ into a single sentence.

```{text}```

```

```
response = get_completion(prompt)
print(response)
```

Providing clear and specific instructions to a model is essential for guiding it towards the desired output and reducing the chances of irrelevant or incorrect responses, with longer prompts often providing more clarity and context for more detailed and relevant outputs.



Tactic 1.2: Structured output

Task: Generate structured outputs

```
In [4]: prompt = """\nGenerate a list of three made-up book titles along \\nwith their authors and genres.\nProvide them in JSON format with the following keys:\\nbook_id, title, author, genre.\n"""\n\nresponse = get_completion(prompt)\nprint(response)
```

```
[{\n    \"book_id\": 1,\n    \"title\": \"The Midnight Garden\",\\n    \"author\": \"Elena Rivers\",\\n    \"genre\": \"Fantasy\"\n}, {\n    \"book_id\": 2,\n    \"title\": \"Echoes of the Past\",\\n    \"author\": \"Nathan Black\",\\n    \"genre\": \"Mystery\"\n}, {\n    \"book_id\": 3,\n    \"title\": \"Whispers in the Wind\",\\n    \"author\": \"Samantha Reed\",\\n    \"genre\": \"Romance\"\n}]
```



Tactic 1.3: Check for conditions

Task: Convert paragraph to line-by-line steps

```
In [5]: text_1 = f"""
Making a cup of tea is easy! First, you need to get some \
water boiling. While that's happening, \
grab a cup and put a tea bag in it. Once the water is \
hot enough, just pour it over the tea bag. \
Let it sit for a bit so the tea can steep. After a \
few minutes, take out the tea bag. If you \
like, you can add some sugar or milk to taste. \
And that's it! You've got yourself a delicious \
cup of tea to enjoy.
"""

prompt = f"""
You will be provided with text delimited by triple quotes.
If it contains a sequence of instructions, \
re-write those instructions in the following format:
```

```
Step 1 - ...
Step 2 - ...
...
Step N - ...
```

If the text does not contain a sequence of instructions, \ then simply write \"No steps provided.\"

```
\\"\\\"{text_1}\\\"\\\""
"""
response = get_completion(prompt)
print("Completion for Text 1:")
print(response)
```

Completion for Text 1:
Step 1 - Get some water boiling.
Step 2 - Grab a cup and put a tea bag in it.
Step 3 - Pour the hot water over the tea bag.
Step 4 - Let the tea steep for a few minutes.
Step 5 - Remove the tea bag.
Step 6 - Add sugar or milk to taste.
Step 7 - Enjoy your delicious cup of tea.

```
In [6]: text_2 = f"""
The sun is shining brightly today, and the birds are \
singing. It's a beautiful day to go for a \
walk in the park. The flowers are blooming, and the \
trees are swaying gently in the breeze. People \
are out and about, enjoying the lovely weather. \
Some are having picnics, while others are playing \
games or simply relaxing on the grass. It's a \
perfect day to spend time outdoors and appreciate the \
beauty of nature.
"""

prompt = f"""
You will be provided with text delimited by triple quotes.
If it contains a sequence of instructions, \
re-write those instructions in the following format:
```

```
Step 1 - ...
Step 2 - ...
...
Step N - ...
```

If the text does not contain a sequence of instructions, \ then simply write \"No steps provided.\"

```
\\"\\\"{text_2}\\\"\\\""
"""
response = get_completion(prompt)
print("Completion for Text 2:")
print(response)
```

Completion for Text 2:
No steps provided.



Tactic 1.4: Few-shot prompting

Task: Answer in a consistent style

```
In [7]: prompt = f"""
Your task is to answer in a consistent style.
<child>: Teach me about patience.

<grandparent>: The river that carves the deepest \
valley flows from a modest spring; the \
grandest symphony originates from a single note; \
the most intricate tapestry begins with a solitary thread.

<child>: Teach me about resilience.
"""

response = get_completion(prompt)
print(response)
```

```
<grandparent>: The tallest trees weather the strongest storms; the brightest stars shine in the darkest nights; the strongest hearts are forged in the hottest fires.
```



Prompting principles

Principle 1: Write clear and specific instructions

Principle 2: Give the model time to “think”



Tactic 2.1: Lay out the steps to complete the task

```
In [8]: text = f"""
In a charming village, siblings Jack and Jill set out on \
a quest to fetch water from a hilltop \
well. As they climbed, singing joyfully, misfortune \
struck-Jack tripped on a stone and tumbled \
down the hill, with Jill following suit. \
Though slightly battered, the pair returned home to \
comforting embraces. Despite the mishap, \
their adventurous spirits remained undimmed, and they \
continued exploring with delight.
"""

# example 1
prompt_1 = f"""

```

Perform the following actions:

- 1 - Summarize the following text delimited by triple \
backticks with 1 sentence.
- 2 - Translate the summary into French.
- 3 - List each name in the French summary.
- 4 - Output a json object that contains the following \
keys: french_summary, num_names.

Separate your answers with line breaks.

```
Text:
```{text}```
```
response = get_completion(prompt_1)
print("Completion for prompt 1:")
print(response)
```

Completion for prompt 1:

1 - Jack and Jill, siblings from a charming village, go on a quest to fet
ch water from a hilltop well, but encounter misfortune along the way.

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

3 - Jack, Jill

4 -

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

Explicit steps vs. CoT

- Action1 → Action2 → ...
- Reason1 → Reason2 → ...

Key Differences

Feature	Explicit Steps in Prompt	Chain-of-Thought (CoT) Prompting
Purpose	Guide step-by-step execution	Encourage step-by-step reasoning
Structure	Direct instructions	Model generates intermediate thoughts
Use Case	Coding, workflows, automation	Math, logic, problem-solving
Example	"Step 1: Do X, Step 2: Do Y"	"Let's think step by step..."

+ Tactic 2.2: Let the model work out its own solution first

Without “thinking”
Answer incorrectly as “True”

```
In [10]: prompt = f"""
Determine if the student's solution is correct or not.
```

Question:
I'm building a solar power installation and I need \ help working out the financials.
- Land costs \$100 / square foot
- I can buy solar panels for \$250 / square foot
- I negotiated a contract for maintenance that will cost \ me a flat \$100k per year, and an additional \$10 / square \ foot
What is the total cost for the first year of operations as a function of the number of square feet.

Student's Solution:
Let x be the size of the installation in square feet.

Costs:
1. Land cost: $100x$
2. Solar panel cost: $250x$
3. Maintenance cost: $100,000 + 100x$
Total cost: $100x + 250x + 100,000 + 100x = 450x + 100,000$
"""\br/>response = get_completion(prompt)
print(response)

The student's solution is correct. The total cost for the first year of operations as a function of the number of square feet is indeed $450x + 100,000$.

With “thinking”
Answer correctly as “False”

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

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

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

```
Use the following format:
Question:
...
question here
...
Student's solution:
...
student's solution here
...
Actual solution:
...
steps to work out the solution and your solution here
...
Is the student's solution the same as actual solution \
just calculated:
...
yes or no
...
Student grade:
...
correct or incorrect
```

Question:

I'm building a solar power installation and I need help \ working out the financials.
- Land costs \$100 / square foot
- I can buy solar panels for \$250 / square foot
- I negotiated a contract for maintenance that will cost \ me a flat \$100k per year, and an additional \$10 / square \ foot
What is the total cost for the first year of operations \ as a function of the number of square feet.

Student's solution:

...

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

Costs:

- Land cost: $100x$
- Solar panel cost: $250x$
- Maintenance cost: $100,000 + 100x$

Total cost: $100x + 250x + 100,000 + 100x = 450x + 100,000$

Actual solution:

"""\br/>response = get_completion(prompt)
print(response)

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

Costs:

- Land cost: $100x$
- Solar panel cost: $250x$
- Maintenance cost: $100,000 + 10x$

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

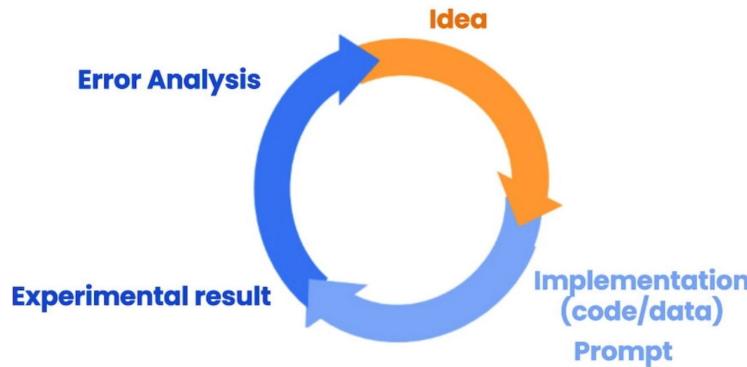
Is the student's solution the same as actual solution just calculated:
No

Student grade:
Incorrect



Iterative Prompt Development

There is no “the perfect prompts” that works with every application. So it is very important to learn how to develop a prompt that works for your specific application.



Prompt guidelines

- Be clear and specific
- Analyze why result does not give desired output.
- Refine the idea and the prompt
- Repeat



Example (1)

The Task

Generate a marketing product description from a product fact sheet

```
In [3]: fact_sheet_chair = """
OVERVIEW
- Part of a beautiful family of mid-century inspired office furniture,
including filing cabinets, desks, bookcases, meeting tables, and more.
- Several options of shell color and base finishes.
- Available with plastic back and front upholstery (SWC-100)
or full upholstery (SWC-110) in 10 fabric and 6 leather options.
- Base finish options are: stainless steel, matte black,
gloss white, or chrome.
- Chair is available with or without armrests.
- Suitable for home or business settings.
- Qualified for contract use.

CONSTRUCTION
- 5-wheel plastic coated aluminum base.
- Pneumatic chair adjust for easy raise/lower action.

DIMENSIONS
- WIDTH 53 CM | 20.87"
- DEPTH 51 CM | 20.08"
- HEIGHT 80 CM | 31.50"
- SEAT HEIGHT 44 CM | 17.32"
- SEAT DEPTH 41 CM | 16.14"

OPTIONS
- Soft or hard-floor caster options.
- Two choices of seat foam densities:
medium (1.8 lb/ft3) or high (2.8 lb/ft3)
- Armless or 8 position PU armrests

MATERIALS
SHELL BASE GLIDER
- Cast Aluminum with modified nylon PA6/PA66 coating.
- Shell thickness: 10 mm.

SEAT
- HD36 foam

COUNTRY OF ORIGIN
- Italy
"""

fact_sheet_chair
```

The Prompt

```
In [4]: prompt = f"""
Your task is to help a marketing team create a
description for a retail website of a product based
on a technical fact sheet.
```

Write a product description based on the information provided in the technical specifications delimited by triple backticks.

```
Technical specifications: """{fact_sheet_chair}"""
"""
response = get_completion(prompt)
print(response)
```

Introducing our stylish and versatile mid-century inspired office chair, perfect for both home and business settings. This chair is part of a beautiful family of office furniture that includes filing cabinets, desks, bookcases, meeting tables, and more.

Customize your chair with several options of shell color and base finishes to suit your personal style. Choose between plastic back and front upholstery or full upholstery in a variety of fabric and leather options. The base finish options include stainless steel, matte black, gloss white, or chrome. You can also choose to have armrests or go for a sleek armless design.

Constructed with a 5-wheel plastic coated aluminum base, this chair features a pneumatic chair adjust for easy raise/lower action. The dimensions of the chair are as follows: width 53 cm, depth 51 cm, height 80 cm, seat height 44 cm, and seat depth 41 cm.

Personalize your chair even further with options such as soft or hard-floor or caster options, two choices of seat foam densities (medium or high), and the choice between armless or 8 position PU armrests.

Made with high-quality materials, including a cast aluminum shell with modified nylon coating and HD36 foam seat, this chair is designed for durability and comfort. Plus, it is qualified for contract use, making it a reliable choice for any workspace.

Add a touch of Italian elegance to your office with this stylish and functional office chair. Elevate your workspace with our mid-century inspired chair that combines style, comfort, and functionality seamlessly.

Too long!



Example (2)

The Task: make it shorter

Generate a marketing product description from a product fact sheet

```
In [3]: fact_sheet_chair = """
OVERVIEW
- Part of a beautiful family of mid-century inspired office furniture,
including filing cabinets, desks, bookcases, meeting tables, and more.
- Several options of shell color and base finishes.
- Available with plastic back and front upholstery (SWC-100)
or full upholstery (SWC-110) in 10 fabric and 6 leather options.
- Base finish options are: stainless steel, matte black,
gloss white, or chrome.
- Chair is available with or without armrests.
- Suitable for home or business settings.
- Qualified for contract use.

CONSTRUCTION
- 5-wheel plastic coated aluminum base.
- Pneumatic chair adjust for easy raise/lower action.

DIMENSIONS
- WIDTH 53 CM | 20.87"
- DEPTH 51 CM | 20.08"
- HEIGHT 80 CM | 31.50"
- SEAT HEIGHT 44 CM | 17.32"
- SEAT DEPTH 41 CM | 16.14"

OPTIONS
- Soft or hard-floor caster options.
- Two choices of seat foam densities:
medium (1.8 lb/ft3) or high (2.8 lb/ft3)
- Armless or 8 position PU armrests

MATERIALS
SHELL BASE GLIDER
- Cast Aluminum with modified nylon PA6/PA66 coating.
- Shell thickness: 10 mm.

SEAT
- HD36 foam

COUNTRY OF ORIGIN
- Italy
"""

fact_sheet_chair
```

The Prompt

- Limit the number of words/sentences/characters.

```
In [5]: prompt = f"""
Your task is to help a marketing team create a
description for a retail website of a product based
on a technical fact sheet.
```

Write a product description based on the information provided in the technical specifications delimited by triple backticks.

Use at most 50 words.

```
Technical specifications: ```{fact_sheet_chair}```
"""
response = get_completion(prompt)
print(response)
```

Introducing our versatile and stylish mid-century office chair, available in a range of colors and finishes. With adjustable height and comfortable upholstery options, this chair is perfect for both home and business use. Made in Italy with quality materials, it's a perfect blend of form and function.

Too general!



Example (3)

The Task: make it more technical

Generate a marketing product description from a product fact sheet

```
In [3]: fact_sheet_chair = """
OVERVIEW
- Part of a beautiful family of mid-century inspired office furniture,
including filing cabinets, desks, bookcases, meeting tables, and more.
- Several options of shell color and base finishes.
- Available with plastic back and front upholstery (SWC-100)
or full upholstery (SWC-110) in 10 fabric and 6 leather options.
- Base finish options are: stainless steel, matte black,
gloss white, or chrome.
- Chair is available with or without armrests.
- Suitable for home or business settings.
- Qualified for contract use.

CONSTRUCTION
- 5-wheel plastic coated aluminum base.
- Pneumatic chair adjust for easy raise/lower action.

DIMENSIONS
- WIDTH 53 CM | 20.87"
- DEPTH 51 CM | 20.08"
- HEIGHT 80 CM | 31.50"
- SEAT HEIGHT 44 CM | 17.32"
- SEAT DEPTH 41 CM | 16.14"

OPTIONS
- Soft or hard-floor caster options.
- Two choices of seat foam densities:
medium (1.8 lb/ft3) or high (2.8 lb/ft3)
- Armless or 8 position PU armrests

MATERIALS
SHELL BASE GLIDER
- Cast Aluminum with modified nylon PA6/PA66 coating.
- Shell thickness: 10 mm.

SEAT
- HD36 foam

COUNTRY OF ORIGIN
- Italy
"""

fact_sheet_chair
```

The Prompt

```
In [8]: prompt = f"""
Your task is to help a marketing team create a
description for a retail website of a product based
on a technical fact sheet.
```

Write a product description based on the information provided in the technical specifications delimited by triple backticks.

The description is intended for furniture retailers,
so should be technical in nature and focus on the
materials the product is constructed from.

At the end of the description, include every 7-character
Product ID in the technical specification.

Use at most 50 words.

```
Technical specifications: ``{fact_sheet_chair}``"
"""
response = get_completion(prompt)
print(response)
```

Introducing our versatile and stylish office chair, part of a mid-century inspired furniture collection. Choose from a variety of shell colors and base finishes to suit your space. Constructed with a durable aluminum base and high-density foam seat for comfort. Perfect for home or business use.

Product IDs: SWC-100, SWC-110

Almost there



Example (4)

The Task: make it formatted

Generate a marketing product description from a product fact sheet

```
In [3]: fact_sheet_chair = """
OVERVIEW
- Part of a beautiful family of mid-century inspired office furniture,
including filing cabinets, desks, bookcases, meeting tables, and more.
- Several options of shell color and base finishes.
- Available with plastic back and front upholstery (SWC-100)
or full upholstery (SWC-110) in 10 fabric and 6 leather options.
- Base finish options are: stainless steel, matte black,
gloss white, or chrome.
- Chair is available with or without armrests.
- Suitable for home or business settings.
- Qualified for contract use.

CONSTRUCTION
- 5-wheel plastic coated aluminum base.
- Pneumatic chair adjust for easy raise/lower action.

DIMENSIONS
- WIDTH 53 CM | 20.87"
- DEPTH 51 CM | 20.08"
- HEIGHT 80 CM | 31.50"
- SEAT HEIGHT 44 CM | 17.32"
- SEAT DEPTH 41 CM | 16.14"

OPTIONS
- Soft or hard-floor caster options.
- Two choices of seat foam densities:
medium (1.8 lb/ft3) or high (2.8 lb/ft3)
- Armless or 8 position PU armrests

MATERIALS
SHELL BASE GLIDER
- Cast Aluminum with modified nylon PA6/PA66 coating.
- Shell thickness: 10 mm.

SEAT
- HD36 foam

COUNTRY OF ORIGIN
- Italy
"""

fact_sheet_chair
```

The Prompt

In [9]: prompt = f"""

Your task is to help a marketing team create a description for a retail website of a product based on a technical fact sheet.

Write a product description based on the information provided in the technical specifications delimited by triple backticks.

The description is intended for furniture retailers, so should be technical in nature and focus on the materials the product is constructed from.

At the end of the description, include every 7-character Product ID in the technical specification.

After the description, include a table that gives the product's dimensions. The table should have two columns. In the first column include the name of the dimension. In the second column include the measurements in inches only.

Give the table the title 'Product Dimensions'.

Format everything as HTML that can be used in a website. Place the description in a <div> element.

Technical specifications: ```{fact_sheet_chair}``````

```
response = get_completion(prompt)
print(response)
```

In [11]: display(HTML(response))

Introducing our latest addition to the mid-century inspired office furniture collection - the versatile and stylish office chair. This chair is designed to elevate any workspace, whether it be at home or in a business setting. With a variety of shell colors and base finishes to choose from, you can customize this chair to fit your aesthetic perfectly.

The construction of this chair is top-notch, featuring a 5-wheel plastic coated aluminum base for stability and mobility. The pneumatic chair adjust allows for easy raise and lower action, ensuring maximum comfort throughout the day.

Constructed with high-quality materials, the shell base glider is made of cast aluminum with a modified nylon PA6/PA66 coating, providing durability and style. The seat is filled with HD36 foam, offering a comfortable seating experience.

Product ID: SWC-100

Product Dimensions:

WIDTH 20.87"

DEPTH 20.08"

HEIGHT 31.50"

SEAT HEIGHT 17.32"

SEAT DEPTH 16.14"

Perfect!

+

Advanced Prompting Techniques



1) Chain-of-Thought Prompting (NeurIPS-2022)

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei

Xuezhi Wang

Dale Schuurmans

Maarten Bosma

Brian Ichter

Fei Xia

Ed H. Chi

Quoc V. Le

Denny Zhou

Google Research, Brain Team

{jasonwei,dennyzhou}@google.com

10 Jan 2023

Abstract

We explore how generating a *chain of thought*—a series of intermediate reasoning steps—significantly improves the ability of large language models to perform complex reasoning. In particular, we show how such reasoning abilities emerge

[1] <https://arxiv.org/abs/2201.11903>

[2] <https://arxiv.org/abs/2205.11916>



1) Chain-of-Thought Prompting (NeurIPS-2022)

- Chain-of-Thought (CoT) prompting enables complex **reasoning capabilities** through intermediate reasoning steps. [1]
- For zero-shot CoT, simply adding “**Let’s think step by step**” also works. [2]

(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. ✓

[1] <https://arxiv.org/abs/2201.11903>

[2] <https://arxiv.org/abs/2205.11916>



2) Self-Consistency (ICLR-2023)

Published as a conference paper at ICLR 2023

SELF-CONSISTENCY IMPROVES CHAIN OF THOUGHT REASONING IN LANGUAGE MODELS

**Xuezhi Wang^{†‡} Jason Wei[†] Dale Schuurmans[†] Quoc Le[†] Ed H. Chi[†]
Sharan Narang[†] Aakanksha Chowdhery[†] Denny Zhou^{†§}**

[†]Google Research, Brain Team

[‡]xuezhiw@google.com, [§]dennyzhou@google.com

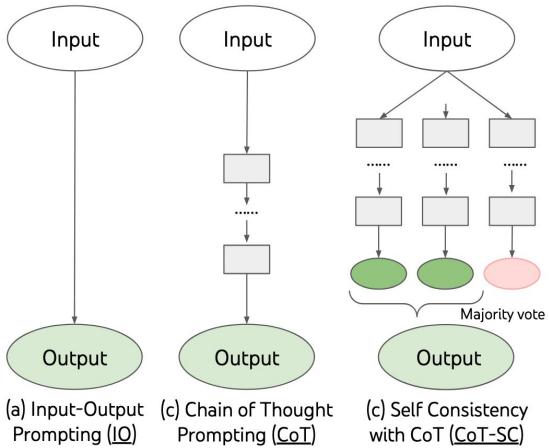
ABSTRACT

Chain-of-thought prompting combined with pre-trained large language models has achieved encouraging results on complex reasoning tasks. In this paper, we propose a new decoding strategy, *self-consistency*, to replace the naive greedy decoding used in chain-of-thought prompting. It first samples a diverse set of reasoning paths instead of only taking the greedy one, and then selects the most consistent answer by marginalizing out the sampled reasoning paths. Self-consistency leverages the



2) Self-Consistency (ICLR-2023)

- Sample multiple, diverse reasoning paths through few-shot CoT,
- and use the generations to select the most consistent answer.



Prompt:

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: We start with 15 trees. Later we have 21 trees. The difference must be the number of trees they planted. So, they must have planted $21 - 15 = 6$ trees. The answer is 6.
Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are $3 + 2 = 5$ cars. The answer is 5.
Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
A: Leah had 32 chocolates and Leah's sister had 42. That means there were originally $32 + 42 = 74$ chocolates. 35 have been eaten. So in total they still have $74 - 35 = 39$ chocolates. The answer is 39.
Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?
A: Jason had 20 lollipops. Since he only has 12 now, he must have given the rest to Denny. The number of lollipops he has given to Denny must have been $20 - 12 = 8$ lollipops. The answer is 8.
Q: Shaw has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?
A: He has 5 toys. He got 2 from mom, so after that he has $5 + 2 = 7$ toys. Then he got 2 more from dad, so in total he has $7 + 2 = 9$ toys. The answer is 9.
Q: There were nine computers in the server room. Five more computers were installed each day, from monday to thursday. How many computers are now in the server room?
A: There are 4 days from monday to thursday. 5 computers were added each day. That means in total $4 * 5 = 20$ computers were added. There were 9 computers in the beginning, so now there are $9 + 20 = 29$ computers. The answer is 29.
Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday, he lost 2 more. How many golf balls did he have at the end of wednesday?
A: Michael initially had 58 balls. He lost 23 on Tuesday, so after that he has $58 - 23 = 35$ balls. On Wednesday he lost 2 more so now he has $35 - 2 = 33$ balls. The answer is 33.
Q: Olivia has \$23. She bought five bagels for \$3 each. How much money does she have left?
A: She bought 5 bagels for \$3 each. This means she spent $5 * 3 = 15$. She has $23 - 15 = 8$ dollars left.
Q: When I was 6 my sister was half my age. Now I'm 70 how old is my sister?
A:

Output 1:

When I was 6 my sister was half my age, so she was 3. Now I am 70, so she is $70 - 6 = 64$. The answer is 64.

Output 2:

When the narrator was 6, his sister was half his age, which is 3. Now that the narrator is 70, his sister would be $70 - 6 = 64$ years old. The answer is 64.

Output 3:

When I was 6 my sister was half my age, so she was 3. Now I am 70, so she is $70/2 = 35$. The answer is 35.



3) ReAct Prompting (ICLR-2023)

Published as a conference paper at ICLR 2023

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao^{*1}, Jeffrey Zhao², Dian Yu², Nan Du², Izhak Shafran², Karthik Narasimhan¹, Yuan Cao²

¹Department of Computer Science, Princeton University

²Google Research, Brain team

¹{shunuy, karthikn}@princeton.edu

²{jeffreyzhao, dianyu, dunan, izhak, yuancao}@google.com

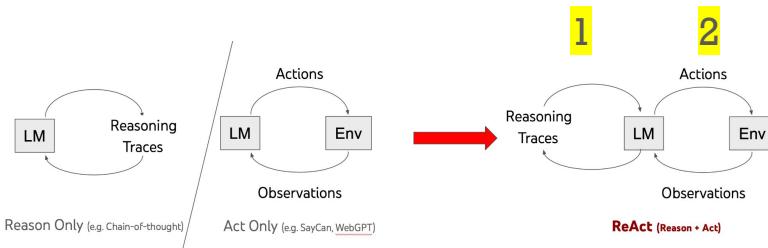
3 [cs.CL] 10 Mar 2023

ABSTRACT

While large language models (LLMs) have demonstrated impressive performance across tasks in language understanding and interactive decision making, their abilities for reasoning (e.g. chain-of-thought prompting) and acting (e.g. action plan generation) have primarily been studied as separate topics. In this paper, we explore the use of LLMs to generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two: reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions allow it to interface with and gather additional information from external sources such as knowledge bases or environments. We apply our approach, named ReAct, to a diverse set of language and decision making tasks and demonstrate its effectiveness over state-of-the-art baselines in addition to improved human interpretability and trustworthiness. Concretely, on question answering (HotpotQA) and fact verification (Fever), ReAct overcomes prevalent

+ 3) ReAct Prompting (ICLR-2023)

- ReAct = Reasoning + Acting
- Allows LLMs perform dynamic reasoning to **create, maintain, and adjust plans for acting** while also enabling interaction with external environments.
- Unlocks agentic capability.



Aside from the Apple Remote, what other devices can control the program, which the Apple Remote was originally designed to interact with?

(1d) ReAct (Reason + Act)

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

Act 1: `Search[Apple Remote]`

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

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

Act 2: `Search[Front Row]`

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

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

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

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

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

Act 4: `Finish[keyboard function keys]`

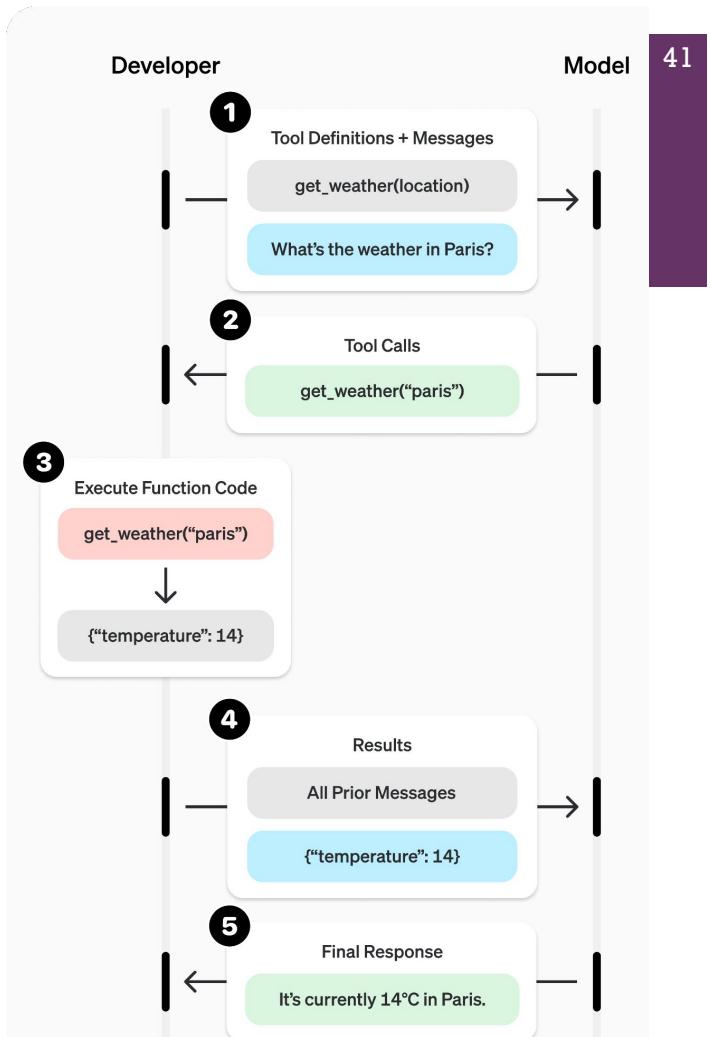


4) Tool Use

LLMs nowadays can also use tools (or functions) to improve the quality of their responses when prompted.

Tools can be anything from:

- Web search
- Code interpreter
- And many more



ToolFormer (MetaAI, Feb 2023)

Toolformer: Language Models Can Teach Themselves to Use ToolsTimo Schick Jane Dwivedi-Yu Roberto Dessì[†] Roberta Raileanu

Maria Lomeli Luke Zettlemoyer Nicola Cancedda Thomas Scialom

Meta AI Research [†]Universitat Pompeu Fabra**Abstract**

Language models (LMs) exhibit remarkable abilities to solve new tasks from just a few examples or textual instructions, especially at scale. They also, paradoxically, struggle with basic functionality, such as arithmetic or factual lookup, where much simpler and smaller models excel. In this paper, we show that LMs can teach themselves to *use external tools* via simple APIs and achieve the best of both worlds. We introduce *Toolformer*, a model trained to decide which APIs to call, when to

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.



Tool Use Example 1

ToolFormer (MetaAI, Feb 2023)

Finetuned a GPT-J model (old model, 6B param) to use tools.

Result: Outperforms GPT-3 (old model, 175B)

Model	ASDiv	SVAMP	MAWPS
GPT-J	7.5	5.2	9.9
GPT-J + CC	9.6	5.0	9.3
Toolformer (disabled)	14.8	6.3	15.0
Toolformer	40.4	29.4	44.0
OPT (66B)	6.0	4.9	7.9
GPT-3 (175B)	14.0	10.0	19.8

Table 4: Results for various benchmarks requiring mathematical reasoning. Toolformer makes use of the calculator tool for most examples, clearly outperforming even OPT (66B) and GPT-3 (175B).

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.



Tool Use Example 2

Program of Thought (PoT) (Google Research, Oct 2023)

Published in Transactions on Machine Learning Research (10/2023)

Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks

[§]Wenhu Chen

[§]Xueguang Ma

[†]Xinyi Wang

[◦]William W. Cohen

[§]University of Waterloo, Canada

[†]University of California, Santa Barbara, USA

[◦]Google Research, USA

wenhu.chen@uwaterloo.ca

x93ma@uwaterloo.ca

xinyi_wang@ucsb.edu

wcohen@google.com

Reviewed on OpenReview: <https://openreview.net/forum?id=YfZ4ZPt8zd>

Abstract

Recently, there has been significant progress in teaching language models to perform step-by-step reasoning to solve complex numerical reasoning tasks. Chain-of-thoughts prompting (CoT) is the state-of-art method for many of these tasks. CoT uses language models to produce text describing reasoning, and computation, and finally the answer to a question. Here we propose ‘Program of Thoughts’ (PoT), which uses language models (mainly Codex)

Tool Use Example 2

45

Program of Thought (PoT) (Google Research, Oct 2023)

Instead of using CoT, PoT outputs reasoning steps in code instead. It then uses the execution result to answer questions

Question: Ketty saves 20000 dollars to the bank. After three years, the sum with compound interest rate is 1000 dollars more than the sum with simple interest rate. What is the interest rate of the bank?

Assuming the interest rate is x . The sum after two years with simple interest rate is $20000 + x * 20000 * 3 = 20000 + 60000x$. The sum after two years with compound interest rate is $20000 * (1 + x)^3 = 200000 + 60000 * x + 60000x^2 + 20000x^3$. The difference can be written as $60000x^2 + 20000x^3 = 1000$. In order to solve x , we can use the quadratic formula. $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, ..., $x = \frac{(-20000 \pm \sqrt{6160})}{120000}$, $x = -0.051333$.

CoT

↓
-0.051333



```
interest_rate = Symbol('x')
sum_in_two_years_with_simple_interest= 20000 +
interest_rate * 20000 * 3
sum_in_two_years_with_compound_interest = 20000 * (1 +
interest_rate)**3
# Since compound interest is 1000 more than simple interest.
ans = solve(sum_after_in_yeras_with_compound_interest -
sum_after_two_years_in_compound_interest - 1000,
interest_rate)
```

PoT

python SymPy

↓
 $x = 0.24814$



Tool Use Example 3

46

Tool Use - in ChatGPT and Claude APIs

Available tools are provided to the model along with their names, descriptions, and schemas in the system prompt.

```
tools = [
    {
        "name": "calculator",
        "description": "A simple calculator that performs basic arithmetic operations.",
        "input_schema": {
            "type": "object",
            "properties": {
                "expression": {
                    "type": "string",
                    "description": "The mathematical expression to evaluate (e.g., '2 + 3 * 4')."
                }
            },
            "required": ["expression"]
        }
    }
]
```



Tool Use - in ChatGPT and Claude APIs (cont.)

Given a message, the model will choose the tool and output the parameters required for you to run the tool and return the result back to the model. Finally, it will return you its final response.

```
=====
User Message: What is the result of 1,984,135 * 9,343,116?
=====
```

Initial Response:

Stop Reason: tool_use

Content: [ContentBlock(text='<thinking>\nThe calculator function is the relevant tool to answer this request, since it involves evaluating a mathematical expression.\n\nThe required parameter for the calculator function is:\nexpression: The mathematical expression to evaluate.\n\nThe human has directly provided the full expression to evaluate in their request: "1,984,135 * 9,343,116". This contains all the information needed for the required expression parameter.\n\nSo I have the necessary information to invoke the calculator tool.\n</thinking>', type='text'), ContentBlockToolUse(id='toolu_01V2mzqp5qkB5QucRFjJUJLD', input={'expression': '1984135 * 9343116'}, name='calculator', type='tool_use')]

Tool Used: calculator

Tool Input: {'expression': '1984135 * 9343116'}

Tool Result: 18538003464660

[ContentBlock(text='Therefore, the result of 1,984,135 * 9,343,116 is 18,538,003,464,660.', type='text')]

Final Response: Therefore, the result of 1,984,135 * 9,343,116 is 18,538,003,464,660.

+

Evaluation

- 1) Closed-ended tasks (with ground-truth)
- 2) Open-ended tasks (no ground-truth)



1) Closed-ended tasks (with ground-truth)

For closed-end tasks, where there is a fixed set of answers, you can easily **compare the predictions with the labels you have**.

For example,

- For text classification tasks, you can use accuracy and F1 score.
- For tasks like MT and summarization, refer to the metrics in the MT slide (lecture 7).
- For coding and math tasks, you usually have the final answers/test cases. Just compare/run the test cases.



Closed-ended tasks dataset examples (English)

GPQA: Graduate-Level Google-Proof Q&A Benchmark

Chemistry (general)

A reaction of a liquid organic compound, which molecules consist of carbon and hydrogen atoms, is performed at 80 centigrade and 20 bar for 24 hours. In the proton nuclear magnetic resonance spectrum, the signals with the highest chemical shift of the reactant are replaced by a signal of the product that is observed about three to four units downfield. Compounds from which position in the periodic system of the elements, which are also used in the corresponding large-scale industrial process, have been mostly likely initially added in small amounts?

- A) A metal compound from the fifth period.
- B) A metal compound from the fifth period and a non-metal compound from the third period.
- C) A metal compound from the fourth period.
- D) A metal compound from the fourth period and a non-metal compound from the second period.

AIME: American Invitational Mathematics Examination

Problem

Every morning Aya goes for a 9-kilometer-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of s kilometers per hour, the walk takes her 4 hours, including t minutes spent in the coffee shop. When she walks $s + 2$ kilometers per hour, the walk takes her 2 hours and 24 minutes, including t minutes spent in the coffee shop. Suppose Aya walks at $s + \frac{1}{2}$ kilometers per hour. Find the number of minutes the walk takes her, including the t minutes spent in the coffee shop.

Solution 1

$$\frac{9}{s} + t = 4 \text{ in hours and } \frac{9}{s+2} + t = 2.4 \text{ in hours.}$$

Subtracting the second equation from the first, we get,

$$\frac{9}{s} - \frac{9}{s+2} = 1.6$$

Multiplying by $(s)(s+2)$, we get

$$9s + 18 - 9s = 18 = 1.6s^2 + 3.2s$$

Multiplying by 5/2 on both sides, we get

$$0 = 4s^2 + 8s - 45$$

Factoring gives us

$$(2s - 5)(2s + 9) = 0, \text{ of which the solution we want is } s = 2.5.$$

Substituting this back to the first equation, we can find that $t = 0.4$ hours.

Lastly, $s + \frac{1}{2} = 3$ kilometers per hour, so

$$\frac{9}{3} + 0.4 = 3.4 \text{ hours, or } \boxed{204} \text{ minutes}$$



Closed-ended tasks dataset examples (Thai)

ThaiExam (available on Huggingface)

Stanford University

 Center for Research on Foundation Models  Stanford University Human-Centered Artificial Intelligence

People Report Research Policy Blog

ThaiExam Leaderboard in HELM

Authors: Yifan Mai and Potsawee Manakul and Sitipong Sripaisarnmongkol and Kunat Pipatanakul and Percy Liang

ThaiExam Leaderboard in HELM

In collaboration with SCBX and SCB 10X, we introduce the [ThaiExam leaderboard](#). ThaiExam is a Thai language benchmark derived from standardized examinations in Thailand. It consists of assessments that evaluate general knowledge at the high school level, such as the ONET, TGAT, TPAT-1, and A-Level exams, as well as the IC exam, which assesses financial knowledge among investment professionals. The ThaiExam leaderboard is the first public leaderboard for language models on Thai language scenarios, and features evaluations of leading language models. Like all other HELM leaderboards, the ThaiExam leaderboard provides full prompt-level transparency, and the results can be fully reproduced using the HELM framework. We hope that this leaderboard will encourage further work in Thai language model evaluation.

scb10x/thai_exam

question	a	b	c	d	e	answer
string · classes	string · classes	string · classes	string · classes	string · classes	string · classes	string · classes
ข้อใดกล่าวในถูกต้องเกี่ยวกับ การทดสอบภาษาไทย	แสดงประพฤติดีว่าได้ทั้งแบบ คุณและของบุคคล	เมื่อแสงผ่านเลนส์จะเกิด การ折射สอดกันท าให้ เกิดรูปแบบจาก	แสงอาทิตย์ธรรมชาติเป็น แสงในโพลาไรซ	รุ่งกินน้ำเกิดขึ้นจาก ปฏิกภาระมีหักเหของแสง เพียงอย่างเดียว	เมื่อแสงเคลื่อนที่ผ่านด้วย กลางท างกันจะมีความเร็ว ไม่เท่ากัน	d
ข้อใดกล่าวไม่ถูกต้องเกี่ยวกับ การทดสอบภาษาไทย	แสดงประพฤติดีว่าได้ทั้งแบบ คุณและของบุคคล	เมื่อแสงผ่านเลนส์จะเกิด การ折射สอดกันท าให้ เกิดรูปแบบจาก	แสงอาทิตย์ธรรมชาติเป็น แสงในโพลาไรซ	รุ่งกินน้ำเกิดขึ้นจาก ปฏิกภาระมีหักเหของแสง เพียงอย่างเดียว	เมื่อแสงเคลื่อนที่ผ่านด้วย กลางท างกันจะมีความเร็ว ไม่เท่ากัน	d

Benchmark

ThaiExam is a multiple-choice question answering benchmark based on Thai examination data from the [Typhoon language model](#). The benchmark comprises the following Thai multiple-choice questions:

- ONET:** The Ordinary National Educational Test (ONET) is an examination for Science, Mathematics, Social Studies, and Science.
- IC:** The Investment Consultant (IC) examination, a licensing test for investment professionals in Thailand.
- TGAT:** The Thai General Aptitude Test (TGAT), a national high school examination.
- TPAT-1:** The Thai Professional Aptitude Test 1 (TPAT-1) is a national high school examination requirement in medical schools. This subset contains reasoning and medical knowledge questions.
- A-Level:** An academic knowledge assessment examination that covers general knowledge.



2) Open-ended tasks (no ground-truth)

- Usually requires **a side-by-side comparison**, e.g., **A/B testing** or manual grading.
- You can also use **an automated method (LLM-as-a-judge)** to compare the responses/grade the responses for you.

<https://lmarena.ai/>

gemini-2.0-pro-exp-02-05

Model A



is there a joke here in this image?

Yes, there's a joke in the image. The humor lies in the absurdity of a snowman shopping for clothes. Snowmen are made of snow and don't typically wear clothes (beyond perhaps a hat, scarf, and carrot nose, as this one does). The snowman is looking at winter clothing, seemingly to find more to wear, which would cause melting.

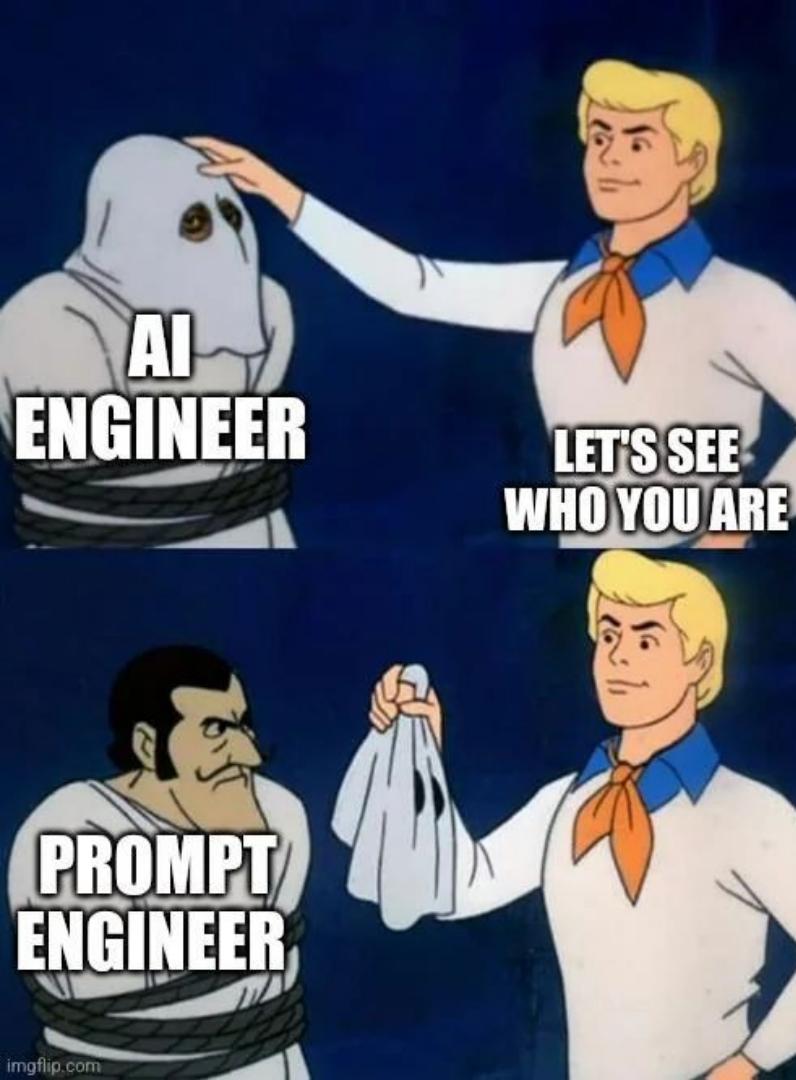
gpt-4o-2024-05-13

Model B



is there a joke here in this image?

Yes, there is a joke in this image. The humor comes from the fact that a snowman is shopping for clothes in a store, which is an unusual and absurd situation because snowmen are made of snow and don't need to wear clothes. The incongruity between the snowman and the typical human activity of shopping for clothes creates the comedic effect.



write a program to find maximum of five inputs from keyboard in python

Here's a [simple Python program](#) that takes **five inputs** from the keyboard and finds the maximum value:

📌 **Basic Version (Using max())**

python

Copy

```
nums = [int(input(f"Enter number {i+1}: ")) for i in range(5)]  
print("Maximum number is:", max(nums))
```

Vibe Coding Is The Future



Y Combinator ✓
1.72M subscribers

Subscribe

Like 3.8K



Share

Download



162,543 views Mar 5, 2025 Lightcone Podcast

Andrej Karpathy recently coined the term "vibe coding" to describe how LLMs are getting so good that devs can simply "give in to the vibes, embrace exponentials, and forget that the code even exists." We dive into this new way of programming and what it means for builders in the age of AI.



Credit to
Aj.Piyalitt