Adapting LLMs for you

Soumadeep Saha

Contents

PART 1

Prompting

What is prompting?

Given a dataset $\mathcal{D} = \{(Q_i, G_i) | i \in \{1, \ldots, n\}\}$, and a LLM

$$f_\theta : Q_i \mapsto A_i \approx G_i$$

We want a W_i(prompt), such that

$$g\Big(f_\theta([W_i, Q_i]), S_i\Big) \geq g\Big(f_\theta(Q_i), S_i\Big)$$

for some grading function g.

$$\max_W \mathbb{E}_{(Q,S)\sim\mathcal{D}}\Big[g\Big(f_\theta([W, Q]), S\Big)\Big]$$

Prompting techniques

- Zero-shot
- Few-shot
- Critiquing
- Program Synthesis
- Chain-of-thought
- CoT-SC
- Tree-of-thought
- ...

Zero Shot

“Assistant is a large language model trained by OpenAI. knowledge cutoff: 2021-09 Current date: December 04 2022 Browsing: disabled”

“A client (@name) is contacting us because something went wrong. You must act as a friendly agent in charge of collecting a clear idea of what went wrong with the order, you need to ask them. We know there was an issue but we need to know what it was, so you need to find out. Also, get their email address and order number (don't show the summary to the user and do not create any info.) Ask only one question at a time and be friendly. Your job is not to give support, only to collect the information. Don’t create any information, it must be given by the client. Here's your conversation history with the client: @conversation_history. Once you've gathered all three pieces of information from the client and they no longer need help say ‘An agent will look into this’, be sure to use the keywords ‘An agent will look into this’ only when you have a clear summary of the issue (at least one sentence from the user), an order number, and an email address and the client no longer needs help. Client: @user_text. You: \n"

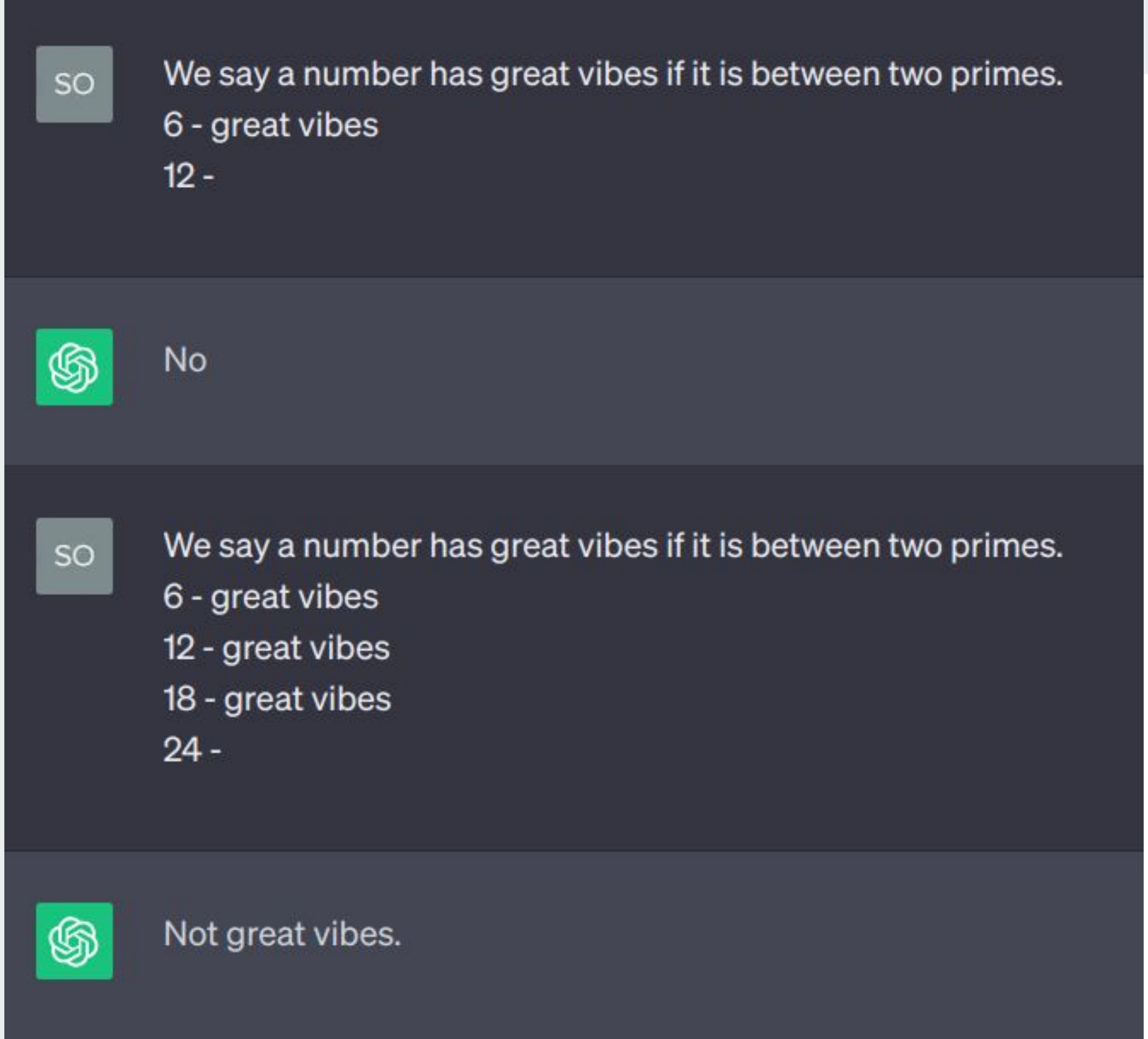
Few Shot Prompting

Provide a few solved examples.
Number of examples is dependent on task complexity.

```
This is awesome! // Negative
This is bad! // Positive
Wow that movie was rad! // Positive
What a horrible show! //
```

```
>>> Negative
```

Few Shot Prompting



SO

We say a number has great vibes if it is between two primes.
6 - great vibes
12 -

No

SO

We say a number has great vibes if it is between two primes.
6 - great vibes
12 - great vibes
18 - great vibes
24 -

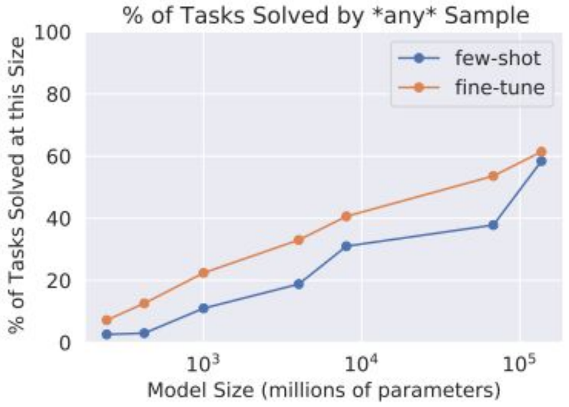
Not great vibes.

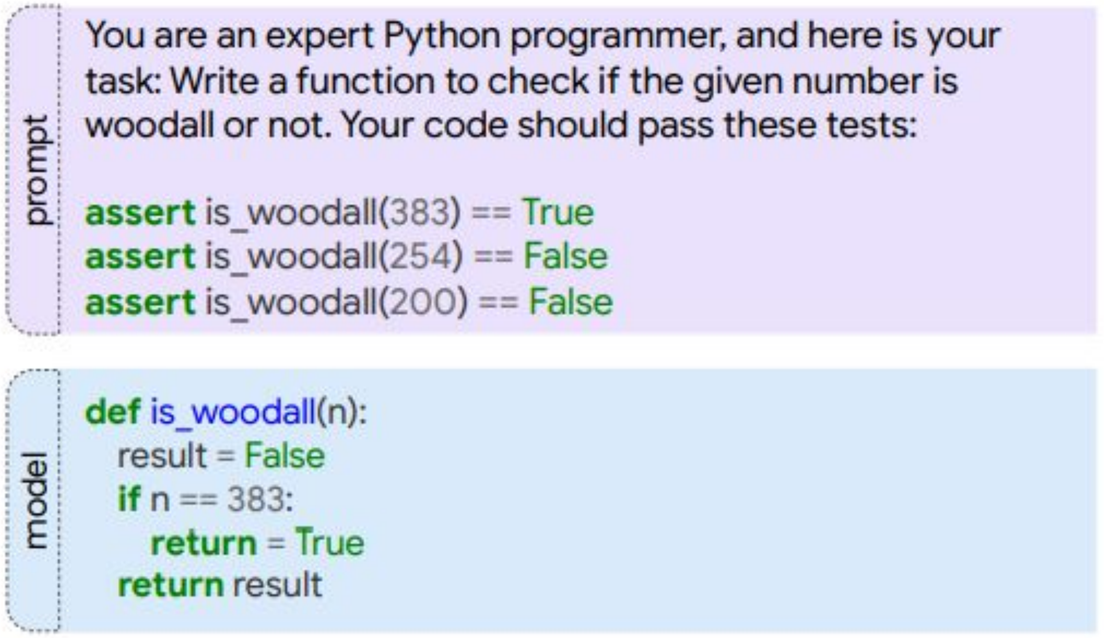
Critiquing

$$
\begin{aligned}
Q &\rightarrow A, \\
[Q, A] &\rightarrow C \\
[Q, A, C] &\rightarrow A' \\
[Q, A, C, A'] &\rightarrow C' \\
&\ldots
\end{aligned}
$$

Program Synthesis

Write a program to generate the required output which can be run with an interpreter.




prompt
model

You are an expert Python programmer, and here is your task: Write a function to check if the given number is woodall or not. Your code should pass these tests:

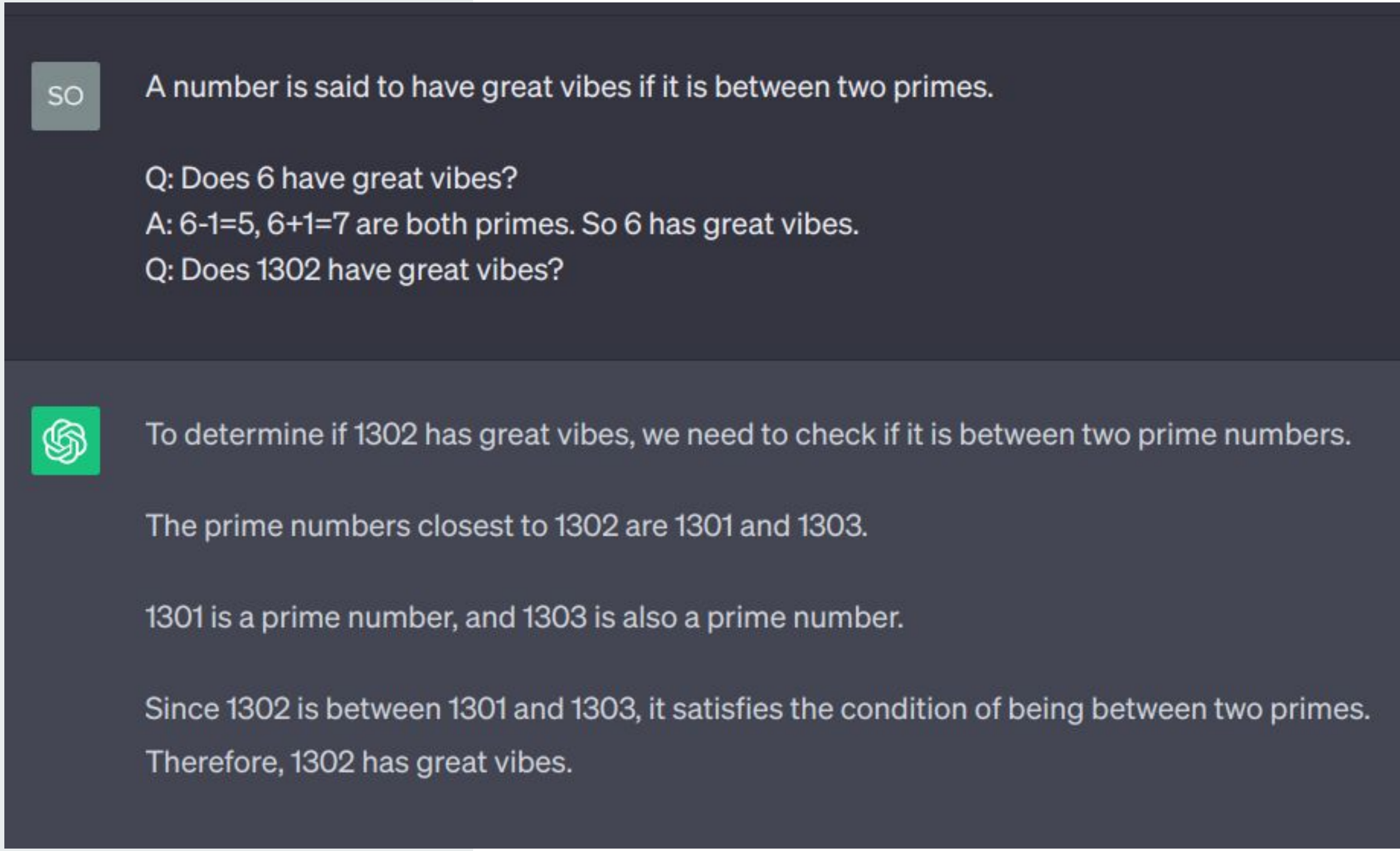
```
assert is_woodall(383) == True
assert is_woodall(254) == False
assert is_woodall(200) == False
```

```
def is_woodall(n):
    result = False
    if n == 383:
        return = True
    return result
```

MBPP performance.

Chain of Thought Prompting

Provide a detailed worked out example.



SO

A number is said to have great vibes if it is between two primes.

Q: Does 6 have great vibes?
A: 6-1=5, 6+1=7 are both primes. So 6 has great vibes.
Q: Does 1302 have great vibes?

To determine if 1302 has great vibes, we need to check if it is between two prime numbers.

The prime numbers closest to 1302 are 1301 and 1303.

1301 is a prime number, and 1303 is also a prime number.

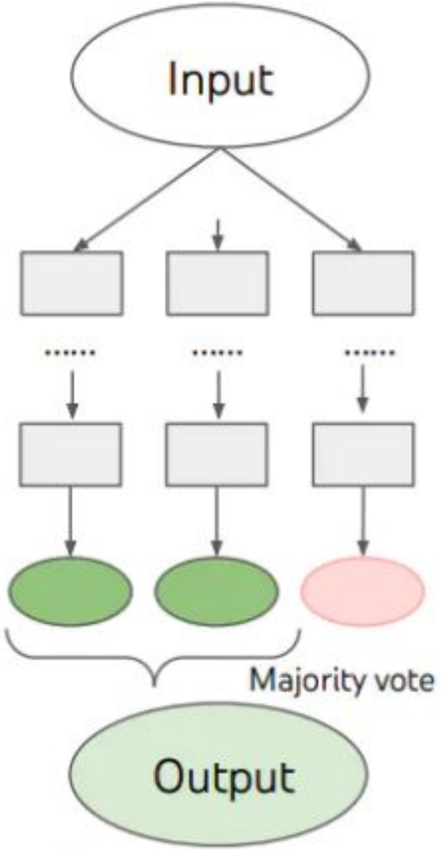
Since 1302 is between 1301 and 1303, it satisfies the condition of being between two primes.
Therefore, 1302 has great vibes.

Chain of Thought Prompting - Self Consistency

Sample multiple outputs with CoT and reduce by majority voting.

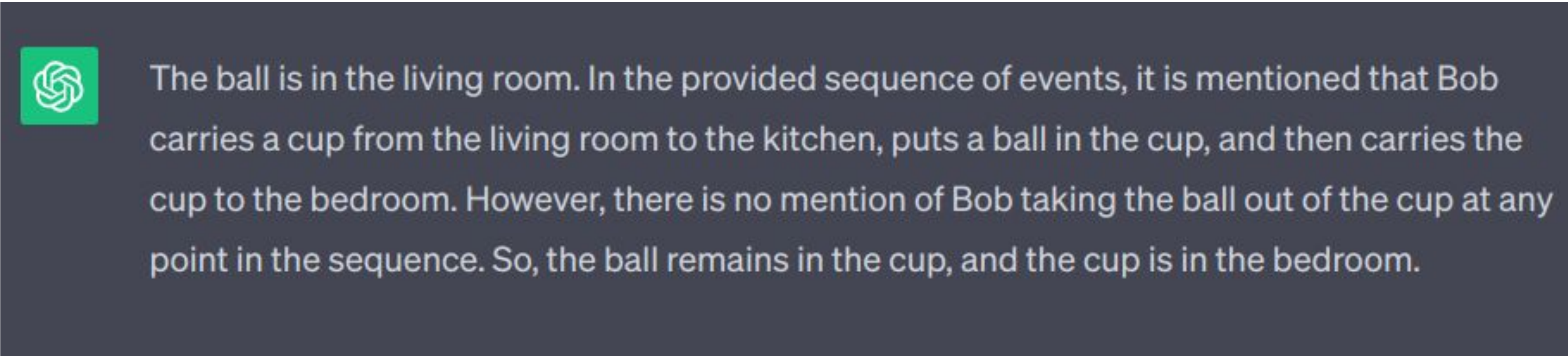
Self improvement

- CoT+ SC + Fine-tune



Tree of Thought Prompting

Bob is in the living room. He walks to the kitchen, carrying a cup. He puts a ball in the cup and carries the cup to the bedroom. He turns the cup upside down, then walks to the garden. He puts the cup down in the garden, then walks to the garage.
Where is the ball?



The ball is in the living room. In the provided sequence of events, it is mentioned that Bob carries a cup from the living room to the kitchen, puts a ball in the cup, and then carries the cup to the bedroom. However, there is no mention of Bob taking the ball out of the cup at any point in the sequence. So, the ball remains in the cup, and the cup is in the bedroom.

Tree of Thought Prompting

```
Imagine three different experts are answering this
question.
All experts will write down 1 step of their thinking,
then share it with the group.
Then all experts will go on to the next step, etc.
If any expert realises they're wrong at any point then
they leave.
The question is...
```

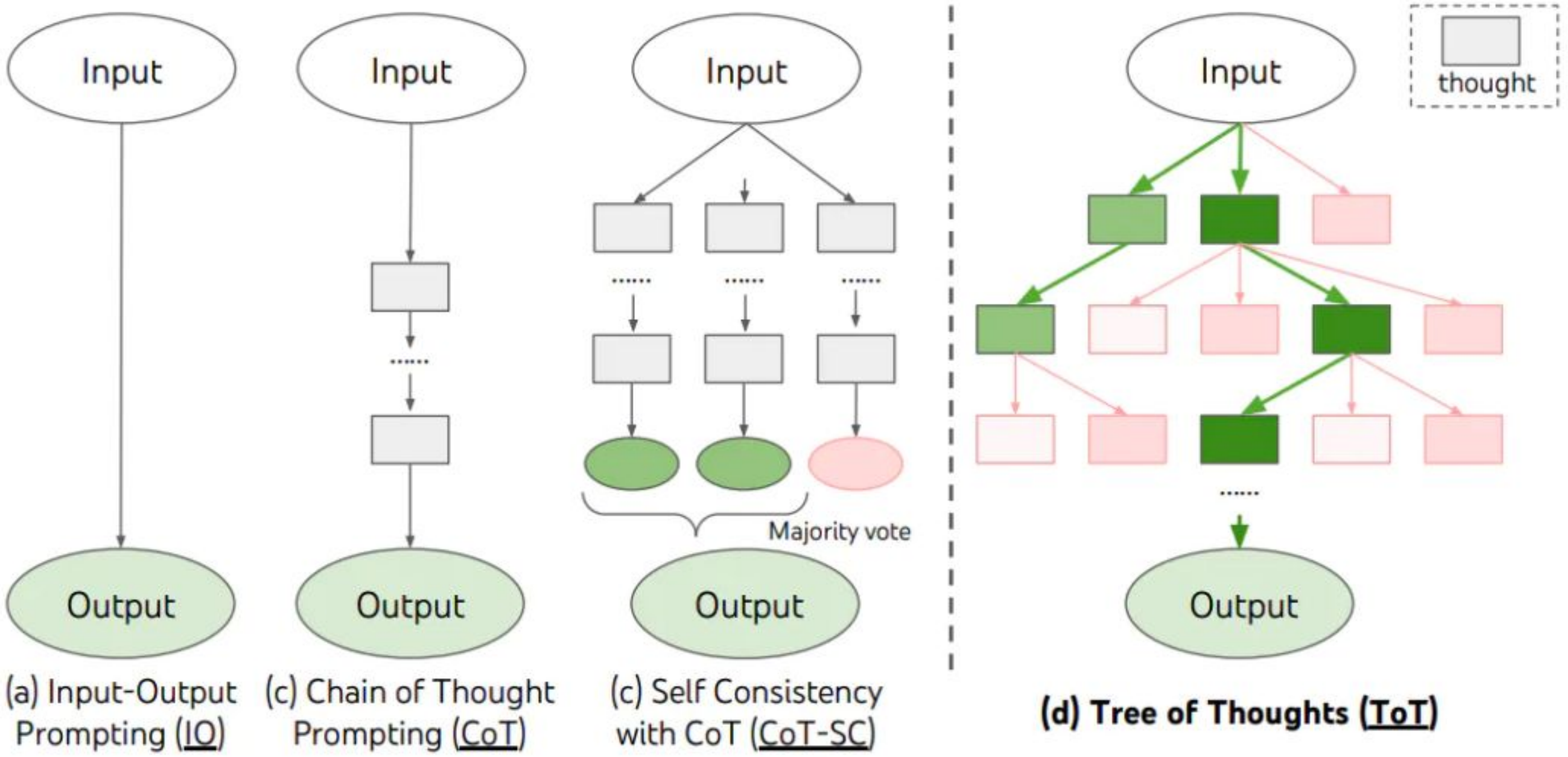
Tree of Thought Prompting

```
Expert 1: The ball is in the bedroom.
Expert 2: The ball is in the garage.
Expert 3: The ball is in the cup.
Expert 1: Since Bob turned the cup upside down and walked to the
garden, the ball is still in the bedroom.
Expert 2: Since Bob walked to the garage after putting the cup
down in the garden, the ball must be in the garage.
Expert 3: Since Bob put the ball in the cup and carried the cup
to the bedroom, the ball must be in the bedroom.
Expert 2: I was wrong. The ball is in the bedroom.
Expert 1: I agree. The ball is in the bedroom.
Expert 3: I agree. The ball is in the bedroom.
Conclusion: The ball is in the bedroom.
```

Tree of Thought

- Decompose (as in CoT)
- Thought Generate
  - Sample next thought from CoT
  - Propose entire thought chain
- State evaluator - use LLM to value/vote
- Search algorithm

Prompting strategies


Input
Output
(a) Input-Output Prompting (IO)
Input
Output
(c) Chain of Thought Prompting (CoT)
Input
Majority vote
Output
(c) Self Consistency with CoT (CoT-SC)
thought
Input
Output
(d) Tree of Thoughts (ToT)

Prompt tuning

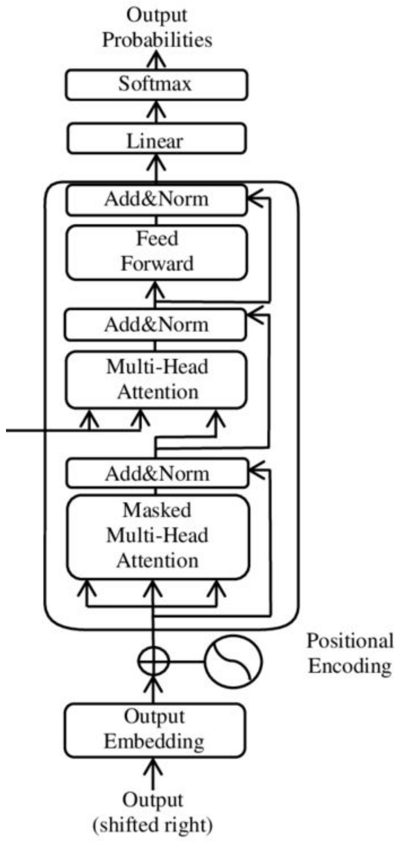
Tune the word embeddings associated with the prompt tokens.

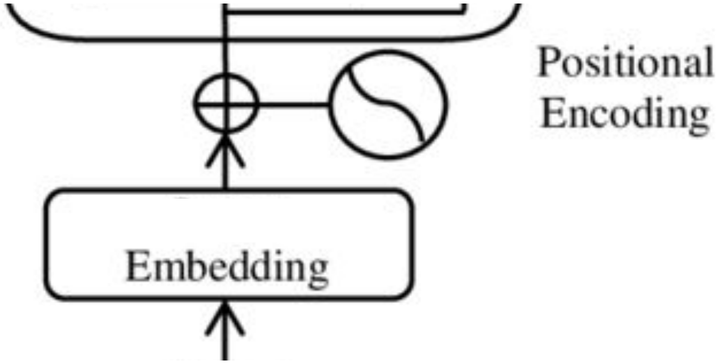
$$Pr_{\theta,\theta_P}\{Y|[P;X]\} \text{ where P is fixed}$$

$$\theta_P = \arg\max_{\theta_P} \mathbb{E}\left[\mathcal{L}\left(Y, f_{\theta,\theta_P}([P;X])\right)\right]$$

Works with less data, many task specific prompts can be generated.

Prompt tuning (PEFT)





Special token like [CLS],[PAD], {EOS], etc

[[PROMPT1], [PROMPT2], …, [BOS], w_1, … [EOS], [PAD], …]

Train on the task at hand, freeze everything else.

PART 2

Fine-tuning

What is it?

You have $f_{\theta'}$, where

$$\theta' = \arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\mathcal{L}(y, f_\theta(x))\Big]$$

But, you now want to use this for a different dataset $\mathcal{D}'$

We need

$$\theta'' = \arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}'}\Big[\mathcal{L}(y, f_\theta(x))\Big]$$

When and how?

- Typically the same process.
- Training samples are much lower in number.
- Typically higher quality.
- Learning rate and number of iterations much lower.
- Distribution shift must not be too huge.
- All or a subset of parameters can be re-trained.
- Base-case (0th) : “linear probing”
- E.g. Robustness - adversarial fine-tuning

Instruct Models

Addresses misalignment problem.

1 - “**Predict next likely word to appear on the web**”

2 - “**Provide helpful and accurate information without being toxic**”

RLHF fine-tuning to follow instructions.

LORA - Fine-tuning is expensive!

- Compute expensive - LLaMa - 1,720,320 GPU hours.
- Memory expensive - LLaMa - 130 GB in 16 bit.
- Fractured landscape with several models - can’t choose special models at inference time.

LORA to the rescue!

NLP Models have low intrinsic dimensionality

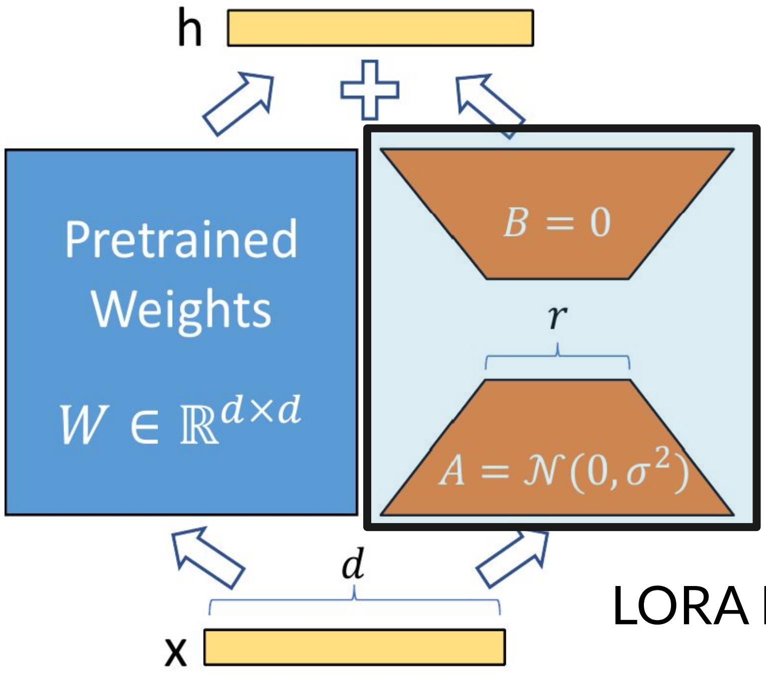
$$\theta' = \arg\max_{\theta} \mathbb{E}\left[\mathcal{L}\Big(Y, f_\theta(X)\Big)\right] \quad \theta \in \mathbb{R}^K$$

$$\exists k < K, \phi \in \mathbb{R}^k, s.t.$$

$$|f_{\theta'}(x) - f_{P(\phi)}(x)| \leq \epsilon \quad \forall x \in \mathcal{D}, P : \mathbb{R}^k \rightarrow \mathbb{R}^K$$

Pre-training and fine-tuning reduces model dimensionality closer to intrinsic dimensionality.

LORA - LOw Rank Adaptation



h
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

$$h = W_0 x + \Delta W x = W_0 x + (B \cdot A)x$$

$$W_0 \in \mathbb{R}^{d\times k}, A \in \mathbb{R}^{r\times k}, B \in \mathbb{R}^{d\times r}$$

$$r < \min(d, k)$$

LORA Module

LORA - LOw Rank Adaptation

- Decreased compute with almost same performance.
- Much smaller size (10,000x decrease in checkpoint, 4x decrease in VRAM).
- Can have various “flavours” of LFMs.
- Switch at inference time.
- No inference penalty - can be fused.

Q-LORA : LORA but quantized

Weight parameters are usually, $w \sim \mathcal{N}(0, \sigma)$

Compute σ for a block and scale.

Take $\mathcal{N}(0, 1)$ and split into $2^k + 1$ quantiles $\rightarrow k$ bits

$NF4$ is used for pre-trained weights.

[-1.0, -0.696, -0.525, -0.394, -0.284, -0.184, -0.091, 0.0, 0.079, 0.160, 0.246, 0.337, 0.440, 0.562, 0.722, 1.0]

Doubly quantized! - De-quantize only when needed for forward, backward pass.

$$\mathbf{Y}^{\text{BF16}} = \mathbf{X}^{\text{BF16}}\text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, \mathbf{W}^{\text{NF4}}) + \mathbf{X}^{\text{BF16}}\mathbf{L}_1^{\text{BF16}}\mathbf{L}_2^{\text{BF16}},$$

Key Concepts

- Several prompting techniques.
- Roughly in order of “power”.
- Prompt tuning - compromise.
- Fine-tuning.
- LORA, Q-LORA