

Pre-train, prompt, predict

If you expect a "raw" LLM (e.g., GPT-3) to behave like ChatGPT: you will be disappointed.

- the LLM is trained to "predict the next" token
 - continue the text given in the prompt
- ChatGPT is a raw LLM that has been Fine-Tuned to be
 - A Helpful Assistant: answer questions

There *are* ways to adapt the raw LLM to be more like ChatGPT without Fine-Tuning.

- Respect the LLM training objective. Turn your questions into instances of "text continuation"
- Use In-Context Learning: exemplars that illustrate the behavior of a Helpful Assistant

Here are suggestions on how to get the "expected behavior" out of the LLaMa LLM:

Prompts to get a raw LLM to generate helpful results

🔗 2. Generations are bad!

Keep in mind these models are not finetuned for question answering. As such, they should be prompted so that the expected answer is the natural continuation of the prompt.

Here are a few examples of prompts (from [issue#69](#)) geared towards finetuned models, and how to modify them to get the expected results:

- Do not prompt with "What is the meaning of life? Be concise and do not repeat yourself." but with "I believe the meaning of life is"
- Do not prompt with "Explain the theory of relativity." but with "Simply put, the theory of relativity states that"
- Do not prompt with "Ten easy steps to build a website..." but with "Building a website can be done in 10 simple steps:\n"

To be able to directly prompt the models with questions / instructions, you can either:

- Prompt it with few-shot examples so that the model understands the task you have in mind.
- Finetune the models on datasets of instructions to make them more robust to input prompts.

We've updated `example.py` with more sample prompts. Overall, always keep in mind that models are very sensitive to prompts (particularly when they have not been finetuned).

Attribution: https://github.com/facebookresearch/llama/blob/llama_v1/FAQ.md

This new paradigm of adapting the behavior of a Pre-Trained model *without* further Fine-Tuning is called "Pre-train, Prompt, Predict" (<https://arxiv.org/pdf/2107.13586.pdf>).

In this module, we explore the "art" of constructing prompts to elicit new behaviors.

Prompt engineering

You see how the behavior of the LLM can be affected by the exact form of the prompt.

- the number of exemplars

It is also the case that slightly changing the words (and order of words) affects behavior.

There is a whole literature on creating successful prompts:

- this [website \(https://www.promptingguide.ai/\)](https://www.promptingguide.ai/) gives good advice in crafting a prompt to meet your expectations
- this [paper \(https://arxiv.org/pdf/2107.13586.pdf\)](https://arxiv.org/pdf/2107.13586.pdf) is a systematic survey
- OpenAI provides [helpful examples \(https://platform.openai.com/examples\)](https://platform.openai.com/examples) for prompting.
- [See Appendix G \(https://arxiv.org/pdf/2005.14165.pdf#page=51\)](https://arxiv.org/pdf/2005.14165.pdf#page=51) (pages 50+) for examples of prompts on creating prompts to solve some standard tasks

Chain of thought prompting

Paper: Chain of thought prompting (<https://arxiv.org/pdf/2201.11903.pdf>)

In school, students are often tasked with solving problems involving multiple steps.

LLM's are better at multi-step reasoning tasks when they have been conditioned to answer step by step.

We call this *chain of thought (CoT)* prompting

The **exemplars** used in CoT prompting

- demonstrate step by step reasoning in the expected output

We can see the difference in the exemplar's "Example Output" section

- using "Standard Prompting" (on the left)
- versus using "CoT Prompting" (on the right)
 - the first exemplar (example) demonstrates step-by-step reasoning

Chain of Thought Prompting

Standard Prompting	Chain of thought prompting
<p>Example Input</p> <p>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?</p>	<p>Example Input</p> <p>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?</p>
<p>Example Output</p> <p>A: The answer is 11.</p>	<p>Example Output</p> <p>Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.</p>
<p>Prompt</p> <p>The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?</p>	<p>Prompt</p> <p>The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?</p>
<p>Model Response</p> <p>The answer is 50. ❌</p>	<p>Model Response</p> <p>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. ✅</p>

Attribution: <https://arxiv.org/pdf/2201.11903.pdf>

How does this apply to the case of *zero* exemplars (zero-shot learning) ?


It turns out that step by step reasoning can be elicited


- Just by adding the phrase "[Let's think step by step](https://arxiv.org/pdf/2205.11916.pdf)"
(<https://arxiv.org/pdf/2205.11916.pdf>) to the end of the query

Let's see an example.

Let's ask ChatGPT to solve a multi-step reasoning problem in a zero-shot setting.

As you can see: it comes close, but produces an incorrect answer.


 Each can contains 3 balls. I start with 5 cans. At the end, all are empty except for one can with 2 balls. How many balls did I use

 You used 15 balls.
This can be determined by starting with the number of cans you have (5) and multiplying it by the number of balls in each can (3) to get 15. Then you subtract the number of balls left in the last can (2) from the total number of balls you started with (15) to get your answer.



The reasoning logic is correct, but the math is wrong.

Now, let's run the same query but append a request to answer step-by-step.

 Each can contains 3 balls. I start with 5 cans. At the end, all are empty except for one can with 2 balls. How many balls did I use. LETS THINK STEP BY STEP



Sure, here is one way to think about the problem step by step:



1. Start by identifying the information given in the problem:
 - Each can contains 3 balls
 - I start with 5 cans
 - At the end, all cans are empty except for one can with 2 balls
2. Next, think about what information you need to find:
 - How many balls did I use
3. Now, you can use the information you have to find the number of balls you used.
 - You started with 5 cans, each containing 3 balls, for a total of $5 \text{ cans} * 3 \text{ balls per can} = 15 \text{ balls}$
 - At the end, you have one can with 2 balls remaining, so you must have used $15 \text{ balls} - 2 \text{ balls} = 13 \text{ balls}$

So, you used 13 balls.

The "LETS THINK STEP BY STEP" seems to condition the model into getting the math correct

- Without CoT: answers comes before reasoning
- With CoT: reasoning precedes answer

It is the conditioning of providing the reasoning before the answer that seems to improve the behavior.

Using zero-shot to create new applications

With a little cleverness, one can almost trivially create a new application using a LLM in zero-shot mode

- create the prefix of a prompt describing the task
- append the user input to the prefix to complete the prompt

Here we use ChatGPT (<https://chat.openai.com/chat>) to create an app that summarizes a conversation

- we create a prompt with a "place-holder" (in braces { . . }) for user input

prompt = Summarize the following conversation: {user input}

< 2/2

KE

summarize the following conversation.

Jane Wilde:

Hey all, great to meet everyone. So where are we on this?

Oscar Maine:


Hi Jane. We'll prepare and share an overview deck with your team this week. Please have your colleagues take a look at it.

Jane Wilde:



Thanks Oscar. Do you have a timeline for setting up an integration with our ERP? That would be very helpful.

Save & Submit

Cancel



Jane Wilde and Oscar Maine discuss preparing an overview deck for Jane's team and setting up an integration with their ERP. Oscar plans to share the deck with the team this week.

Here we use ChatGPT as a programming assistant

prompt = Write a Python function that does the following: {task description}



Write a Python program that does the following: add a list of numbers



Here is a sample Python program that adds a list of numbers:

```
Copy code

# Define a list of numbers
numbers = [1, 2, 3, 4, 5]

# Initialize the sum to 0
sum = 0

# Iterate over the numbers in the list
for number in numbers:
    # Add the current number to the sum
    sum += number

# Print the final sum
print(sum)
```


Some more, creative examples

- [Spreadsheet add-in to perform lookups](https://twitter.com/pavtalk/status/1285410751092416513)
(<https://twitter.com/pavtalk/status/1285410751092416513>)
- [Generate a web page from a description](https://twitter.com/sharifshameem/status/1283322990625607681)
(<https://twitter.com/sharifshameem/status/1283322990625607681>)

References found in: <http://ai.stanford.edu/blog/understanding-incontext/>
(<http://ai.stanford.edu/blog/understanding-incontext/>).

How is zero-shot learning possible ? Some theories

Theory 1

- The training set contains explicit instances of these out of sample tasks

Theory 2

- The super-large training sets contain *implicit* instances of these out of sample tasks
 - For example: an English-language article quoting a French speaker in French with English translation

One thing that jumps out from the graph:

- Bigger models are more likely to exhibit meta-learning

Theory 3

The training sets are so big that the model "learns" to create groups of examples with a common theme

- Even with the large number of parameters, the model capacity does not suffice for example memorization

Another thing to consider

- The behavior of an RNN depends on *all* previous inputs
 - It has memory (latent state, etc.)

So Few Shot Learning may work by "priming" the memory with parameters for a specific task

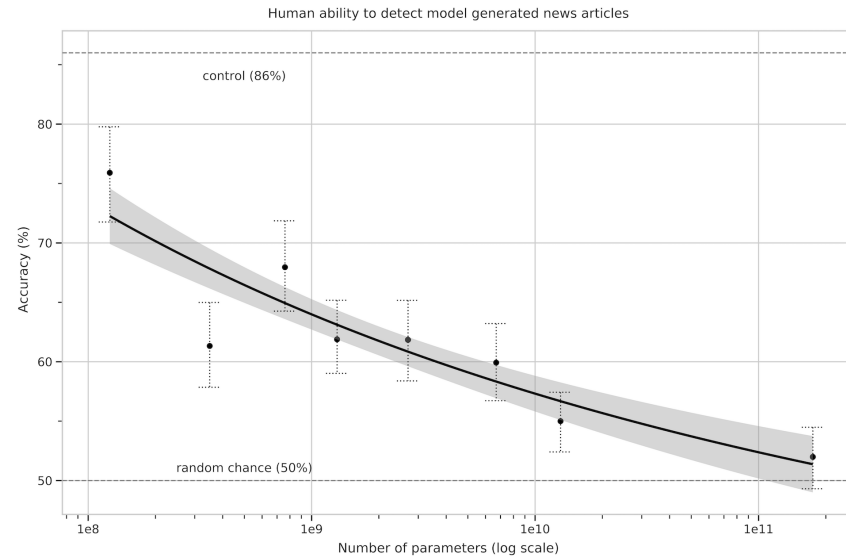
Social concerns

The team behind GPT is very concerned about potential misuse of Language Models.

To illustrate, they conducted an experiment in having a Language Model construct news articles

- Select title/subtitle of a genuine news article
- Have the Language Model complete the article from the title/subtitle
- Show humans the genuine and generated articles and ask them to judge whether the article was written by a human

Human accuracy in detecting model generated news articles



Picture from: <https://arxiv.org/pdf/2005.14165.pdf>

The bars show the range of accuracy across the 80 human judges.

- 86% accuracy detecting articles created by a really bad model (the control)
- 50% accuracy detecting articles created by the biggest models

It seems that humans might have difficulty distinguishing between genuine and generated articles.

The fear is that Language Models can be used

- to mislead
- to create offensive speech

Harmful behavior of the Pre-Trained LLM

Another important concern is that LLM's can exhibit bias

- we saw similar behavior with embeddings
- the bias is statistical: a property of the training data

Any model based on the Pre-Trained model might inherit this bias.

A recent trend in model development

- is to probe the model for harmful behavior

Models are now often described via a [Model card \(https://arxiv.org/abs/1810.03993\)](https://arxiv.org/abs/1810.03993).

- describing both the technical properties
- and the social aspects (biases)

[Here \(https://huggingface.co/openai-gpt\)](https://huggingface.co/openai-gpt) is an example of a model card.

In [2]: `print("Done")`

Done

