

Multi-task learning

One area of recent interesting is *multi-task learning*

- Training a model to implement multiple tasks

A model that implements a single task computes

$$p(\text{output} \mid \text{input})$$

A model that implements several tasks computes

$$p(\text{output} \mid \text{input}, \text{task-id})$$

When training a model for multiple tasks, the training examples would look something like:

(Translate to French, English text, French Text)
(Answer the question, document, question, answer)

Text is almost a universal encoding so NLP is a natural way of expressing multiple tasks.

So a natural extensions of a Language Model is to solve multiple tasks

- Encode your specific task as an input that can be handled by a Language Model
- That's one advantage of Byte Pair Encoding
 - No special per-task pre-processing needed for a task's training set

We will take the idea of Multi-task learning one step further

- Learning how to solve a task **without** explicitly training a model !

Pre-train, prompt, predict

We have presented the "Unsupervised Pre-Training + Supervised Fine-Tuning" paradigm.

Considering that

- Language models seem to learn universal, task-independent language representation
- Text-to-text is a universal API for NLP tasks

We can raise the question

- Is Supervised Fine-Tuning even necessary ?
- Can a Language Model learn to solve a task *without having been trained on examples for the task* ?

There are some practical impediments to answering this question

- How does the LM model "understand" that it is being asked to solve a particular task ?
- How does the LM model "understand" the input-output relationship involved in the new task ?

The solution to both impediments is to *condition the LLM* by pre-pending paradigmatic examples at the start of every query

- we will call this the *pre-prompt*

For example, we can describe Translation between languages with the following pre-prompt

Translate English to French

sea otter => loutre de mer

peppermint => menthe poivree

plush giraffe => girafe peluche

cheese =>

The prompt consists of

- an initial string describing the task: "Translate English to French"
- a number of examples
 - English input, French output, Separated by a =>
- a new example **without** a target, representing a query to be solved

cheese =>

The expectation is that the "next words" generated by the Language Modeling task

- are the translation of cheese into French

Note that the labeled "examples" are given at *inference* time **not** training time

- the model's weights are **not updated**
- the examples only condition the model into generating specific output

This paradigm has been called "[Pre-train, Prompt, Predict](https://arxiv.org/pdf/2107.13586.pdf)"
(<https://arxiv.org/pdf/2107.13586.pdf>).

More formally:

- Let C ("context") denote the pre-prompt.
- Let \mathbf{x} denote the "query" (e.g., cheese =>)

The LLM implies a probability distribution for the characters it will generate following query \mathbf{x}

$$p(\mathbf{x})$$

By pre-pending the context C , we have conditioned the LLM to produce conditional distribution

$$p(\mathbf{x}|C)$$

Zero shot learning: learning to learn

Let k denote the number of labeled examples in the pre-prompt.

We can ask how well a LLM performs on an unknown query with varying size of k .

- **Few shot learning:** $10 \leq k \leq 100$ typically
- **One shot learning:** $k = 1$
- **Zero shot learning** $k = 0$

A picture will help

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.



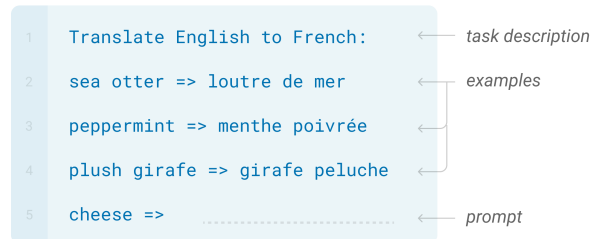
One-shot

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

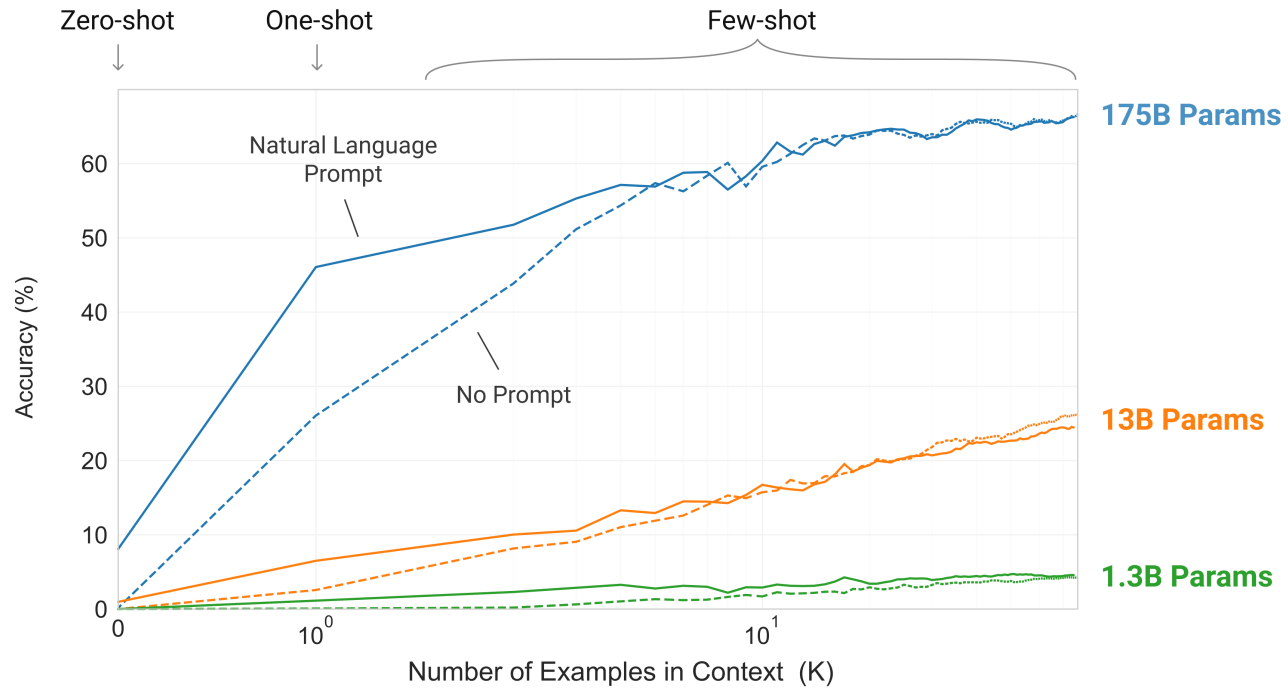
The model is trained via repeated gradient updates using a large corpus of example tasks.



Is this even possible ?! Learning a new task with **zero** examples ?

Let's look at the reported results from the third generation GPT-3 model.

Few/One/Zero shot learning



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

Prompt engineering

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

There is a whole literature on creating successful prompts: [Prompt engineering](https://arxiv.org/pdf/2107.13586.pdf) (<https://arxiv.org/pdf/2107.13586.pdf>), [Chain of thought prompting](https://arxiv.org/pdf/2201.11903.pdf) (<https://arxiv.org/pdf/2201.11903.pdf>).

- Providing enough context to condition the model to "understand"
 - That "Translate English to French" relates to some examples seen (implicitly) in training
 - and that the string => suggests a relationship between the input and output
 - perhaps generalizing examples seen in training

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 for many other tasks.

Chain of thought prompting

[Paper: Chain of thought prompting \(https://arxiv.org/pdf/2201.11903.pdf\)](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 prompting"

- the prompt describes the reasoning process needed for one or more examples
- and finishes with a query (example with no reasoning)

Standard Prompting

Example Input

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

Example Output

A: The answer is 11.

Prompt

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

Model Response

The answer is 50.



Chain of thought prompting

Example Input

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

Example Output

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

Prompt

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

Model Response

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.



The example was using "few-shot Chain of Thought (CoT)" prompting.

It turns out: "**zero-shot** Chain of Thought" prompting works !

- Just add 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 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.

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 $5 \text{ cans} \times 3 \text{ balls per can} = 15 \text{ balls}$
 - 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.

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 >  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)
```

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

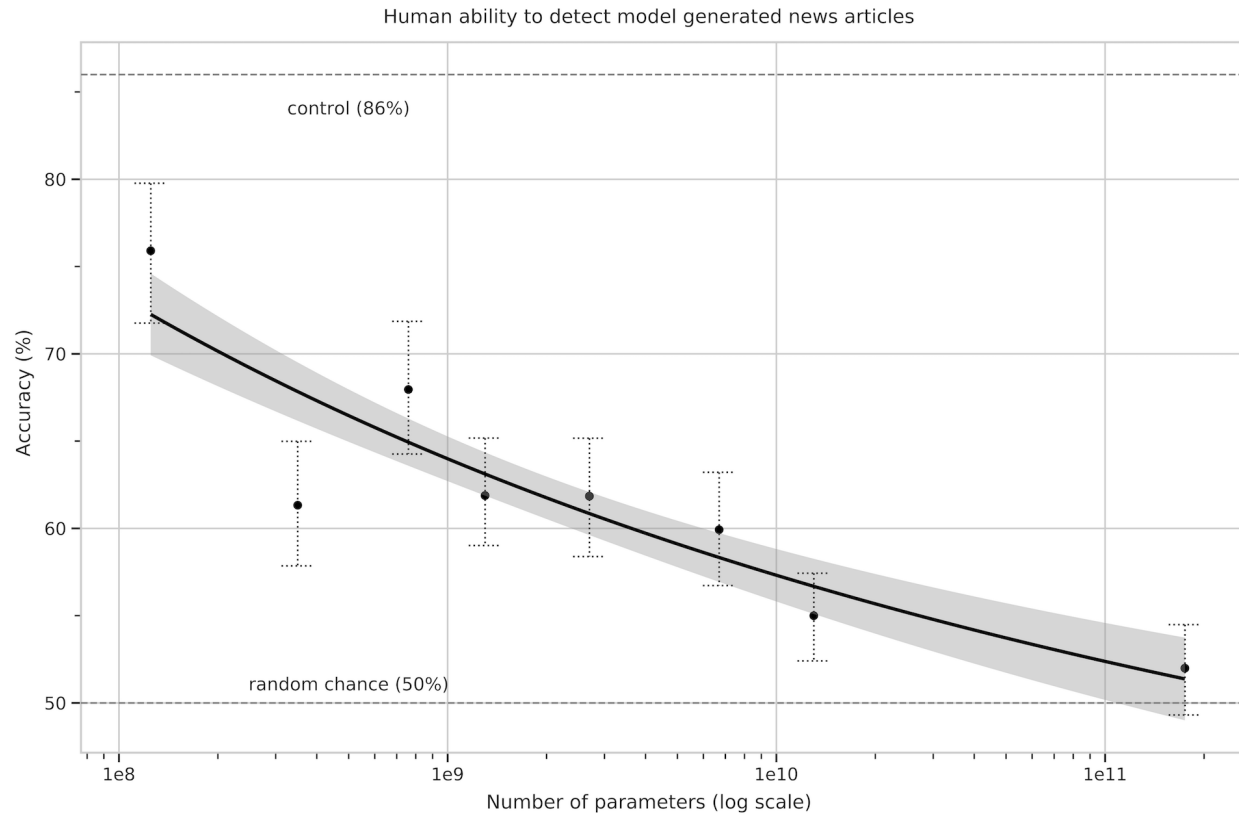
Social concerns

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

```
In [1]: print("Done")
```

Done

