In-Context Learning

The Universal API also opens up an interesting possibility

- Can we use a Large Language Model
- To solve a new Target task
- *without* further training (i.e., Fine-Tuning)

On a pure syntax level, this is feasible

- all problems are instances of "predict the next"
- perhaps the LLM's text completion power *also* captures domain-specific knowledge
    - the completion is related to the domain-specific words of the prompt
    - for example, if the prompt is in French, the completion would be expected to be in French too

So the possibility is there.

But how does the Source LLM discern what the Target task is ?

The answer

- we provide demonstrations (*examplars*) of the new task as part of the *Context*
    - demonstrating the relationship between Input and Output
    - as the initial part of the prompt
- and present the Input for a test example as the final part of the prompt
    - no output
    - hope "predict the next" will result in the correct Output

For example, we can describe Translation between languages with the following Context C

```
Translate English to French

sea otter =>  loutre de mer

peppermint => menthe poivree

plush giraffe => girafe peluche
```

The expectation is that when the user presents the prompt $\mathbf{x}$

```
cheese =>
```

the model will respond with the French translation of `cheese` .

- the "next words" predicted by the Language Modeling

The idea behind *In-Context Learning* is to

- condition the Pre-Trained Language Model (Source LLM)
- to complete the prompt of a new Target Task
- with the correct response for the Target Task
- **without** further training (Fine-Tuning)
- merely by providing the examples that demonstrate the behavior of the new Target task
- **as parts of the prompt**

The examples demonstrating the Target task are called *exemplars*.

- each exemplar is provided as a prompt and response pair, as per the Universal API

$$\langle \mathbf{x^{(i)}}, \mathbf{y^{(i)}} \rangle = \langle \mathrm{prompt}, \mathrm{response} \rangle$$

To predict the response for a new prompt $\mathbf{x}$

- the exemplars are concatenated together (the *Context C*)
- the Context is a demonstration of the Target Task's relationship between features and label

The new prompt $\mathbf{x}$ is appended to the Context

- the Pre-Trained model is expected to complete the prompt
- by providing a response specific to the Target task and the prompt $\mathbf{x}$

More formally:

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

The unconditional Language Modeling objective
$$p(\mathbf{y}|\mathbf{x})$$
is to create the sequence $\mathbf{y}$ that follows the sequence of prompt $\mathbf{x}$.

Here, the pre-prompt conditions the model's objective
$$p(\mathbf{y}|C, \mathbf{x})$$
to create the sequence $\mathbf{y}$ that follows from the exemplars C and prompt $\mathbf{x}$.

This is just a mechanical process

- create the sequence $\dot{\mathbf{x}}$
- by concatenating some number k of exemplars: $\langle \mathbf{x}^{(1)}, \mathbf{y}^{(1)} \rangle, \ldots, \langle \mathbf{x}^{(k)}, \mathbf{y}^{(k)} \rangle$
- and prompt string $\mathbf{x}$
- delimiting elements by separator characters $\langle \mathrm{SEP}_1 \rangle . \langle \mathrm{SEP}_2 \rangle$

$$\dot{\mathbf{x}} = \mathrm{concat}(\quad \mathbf{x}^{(1)}, \langle \mathrm{SEP}_1 \rangle, \mathbf{y}^{(1)}, \langle \mathrm{SEP}_2 \rangle,$$
$$\vdots$$
$$\mathbf{x}^{(k)}, \langle \mathrm{SEP}_1 \rangle, \mathbf{y}^{(k)}, \langle \mathrm{SEP}_2 \rangle,$$
$$\mathbf{x}$$
$$)$$

The LLM then computes

$$p(\mathbf{y}|\dot{\mathbf{x}})$$

For convenience, we will just write this as the conditional probability

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

In-Context learning: let's experiment

The HuggingFace platform (https://huggingface.co/) has libraries of pre-trained models for many tasks, including Language models.

There is a clean API for using these models in code (I recommend their on-line course (https://huggingface.co/) if you want to play with it).

But they also host many of their models for interactive use.

This is valuable not just for the obvious reason of ease of use
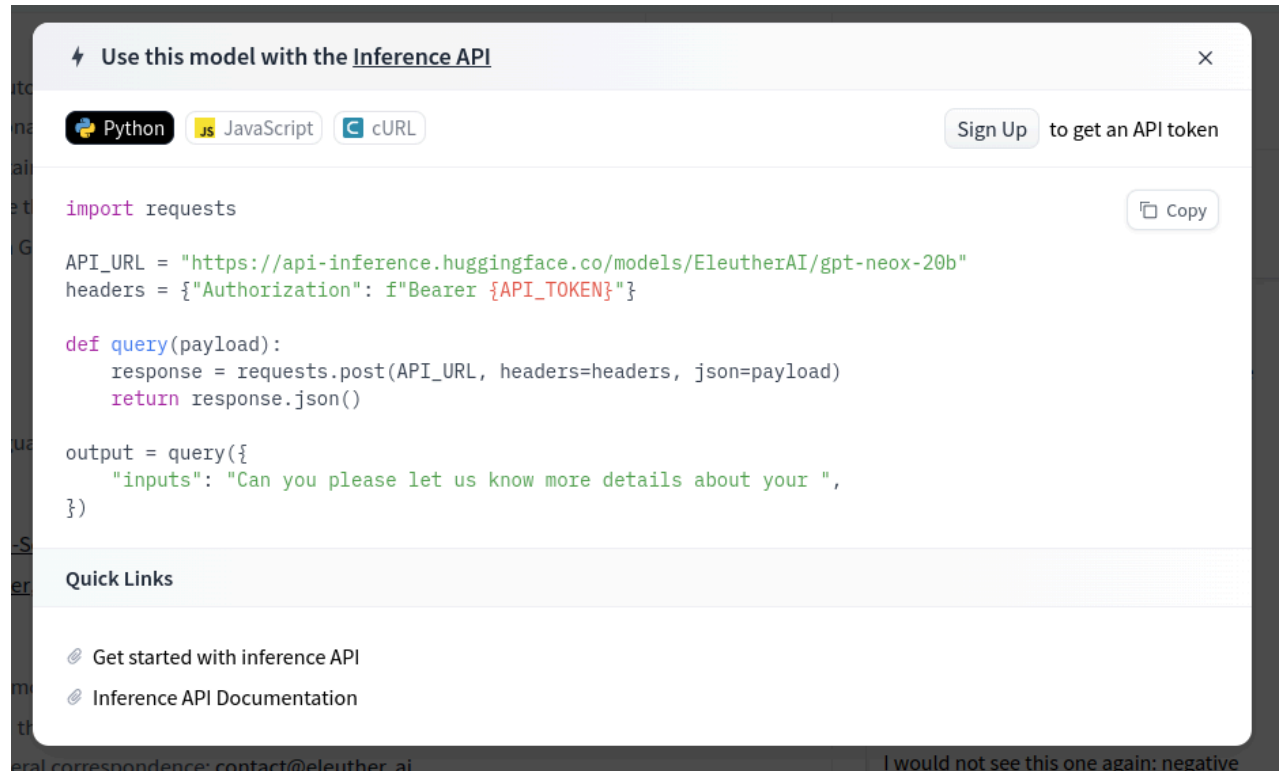
- some models are too big to load on the machines available to us

You can play with In-Context learning by going to the page of a model and typing into the *Hosted Inference API* text box.

But there is also an API that allows you to pass the input (context plus prompt) via a URL.

If you click on the `Deploy` button and choose the `Inference API` drop-down

- you will see Python code for querying the model programaticly.

⚡ **Use this model with the Inference API**   ✕

🐍 Python   JS JavaScript   C cURL   Sign Up to get an API token

```python
import requests                                                    Copy

API_URL = "https://api-inference.huggingface.co/models/EleutherAI/gpt-neox-20b"
headers = {"Authorization": f"Bearer {API_TOKEN}"}

def query(payload):
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.json()

output = query({
    "inputs": "Can you please let us know more details about your ",
})
```

Quick Links

🔗 Get started with inference API

🔗 Inference API Documentation

Please note

- our toy example above used a *single* test example
- even if we manage to get a correct prediction on a single example
    - we don't have confidence that the new task was successfully learned !
    - we really should evaluate success on a larger number of text examples
- still: the fact that the exemplars taught the model the correct syntax for an answer is exciting

Here is a very crude notebook that uses the HuggingFace inference API to experiment with in-context learning.

- [Experiment in In context learning: Colab (https://colab.research.google.com/github/kenperry-public/ML_Advanced_Fall_2024/blob/master/HF_inference_play.ipynb)](https://colab.research.google.com/github/kenperry-public/ML_Advanced_Fall_2024/blob/master/HF_inference_play.ipynb)
- [Experiment in In context learning: local (HF_inference_play.ipynb)](HF_inference_play.ipynb)

Our new Target task is movie reviews.

Here are the exemplars we will use

```
exemplars = [ "this movie was great: positive",
              "one of the best films of the year: positive",
              "just plain awful: negative",
              "I would not see this one again: negative",
              "this movie was great: positive",
              "one of the best films of the year: positive",
              "just plain awful: negative",
              "I would not see this one again: negative",
              "I love this film: positive"

]
```

Not the greatest exemplars (too short), but we just want to illustrate the idea.

We want the model to classify the following review as poistive/negative

```
"I've heard not so great things about this one:"
```

We would hope the exemplars are sufficient to cause the "predict the next" task to generate the continuation

```
negative
```

We will use a relatively small (20B parameters) LLM.

Here are the results

```
[0.22 seconds, using EleutherAI/gpt-neox-20b]
 this movie was great: positive
 one of the best films of the year: positive
 just plain awful: negative
 I would not see this one again: negative
 this movie was great: positive
 one of the best films of the year: positive
 just plain awful: negative
 I would not see this one again: negative
 I love this film: positive
 I've heard not so great things about this one: negative
```

Negative ! Just as we had hoped.

However, the exemplars are not ideal. The LLM continues *beyond* the classification and further generates more reviews !

it is entertaining: positive
I would not see this one again: negative
I love this film: positive
I've heard not so great things: negative
it is entertaining: positive
I've heard not so great things: negative
enchanced by slow motion visuals: positive
excellent: positive
terrific sound design in this one too: positive
Stallone is a great actor: positive
I'd turn down a free trip to London to see this movie:

Obviously, our exemplars did not fully convey our intent.

Creating a prompt (context) that will cause an LLM to do *exactly* what you want

- is an art
- called Prompt Engineering

Learning to learn

Does In-Context learning really work ?

We can begin to answer this question by

- examining the behavior of a Pre-Trained LLM
- on a new task
- using k exemplars
  - varying k

Depending on k, we refer to the behavior of the LLM by slightly different names

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

A picture will help

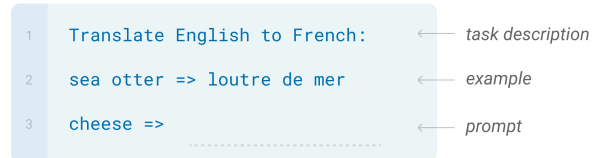## The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.
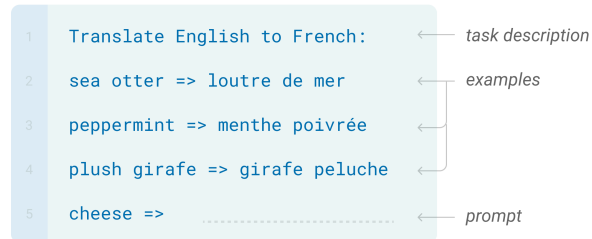
```
1   Translate English to French:        ←—— task description

2   cheese =>          ..................  ←—— prompt
```

One-shot

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

```
1   Translate English to French:        ←—— task description

2   sea otter => loutre de mer          ←—— example

3   cheese =>          ..................  ←—— prompt
```

Few-shot

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

```
1   Translate English to French:        ←—— task description

2   sea otter => loutre de mer          ←—

3   peppermint => menthe poivrée        ←—  examples

4   plush girafe => girafe peluche      ←—

5   cheese =>          ..................  ←—— prompt
```

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.

```
1   sea otter => loutre de mer          ←—— example #1
```
↓
gradient update
↓
```
1   peppermint => menthe poivrée        ←—— example #2
```
↓
gradient update
↓
• • •
↓
```
1   plush giraffe => girafe peluche     ←—— example #N
```

gradient update

```
1   cheese =>          ..................  ←—— prompt
```

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

Let's look at the reported In-Context Learning results of 3 LLM's of varying size.

Picture from: https://arxiv.org/pdf/2005.14165.pdf#page=4
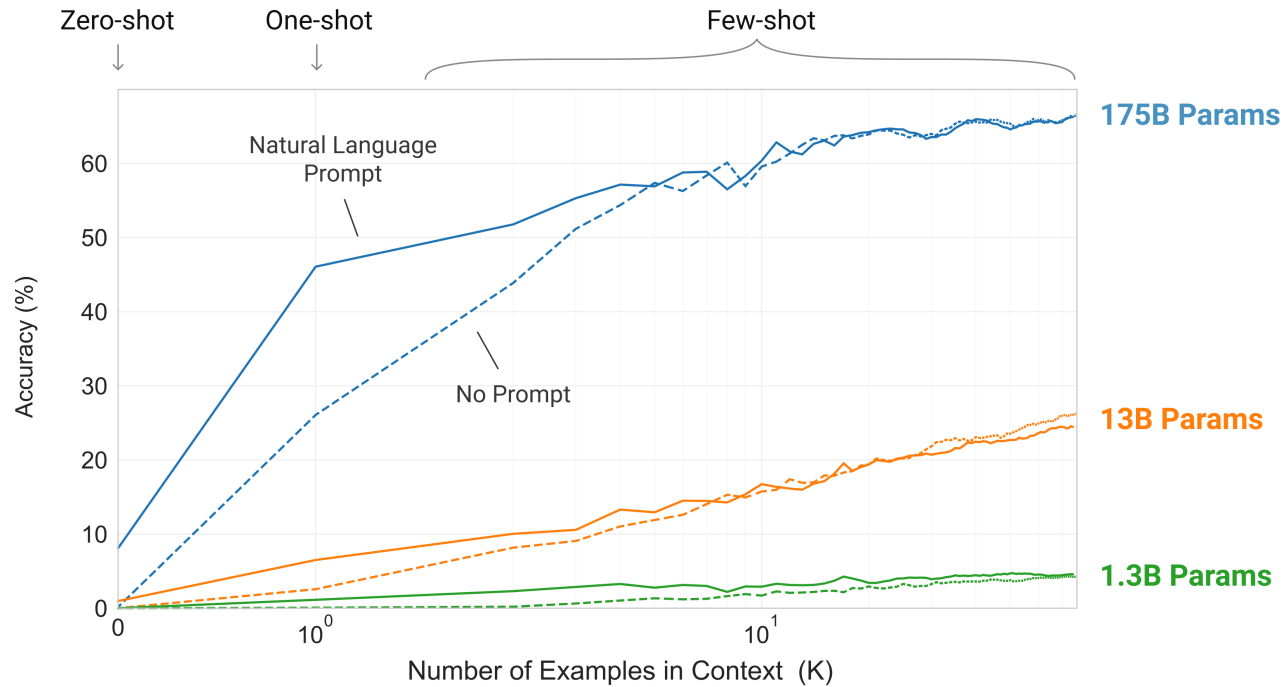
A couple of observations

- As the size of the model grows: In-Context Learning behavior improves
    - compare the 175 Billion parameter model to the smaller models
    - we sometimes refer to this as behavior that "emerges" only when a model is sufficiently large

- More exemplars (greater k) helps

    - but not much for the smallest model

- Zero shot learning works !

    - but this is a behavior that only emerges for very large models

"Fine-tuning" a model with In-Context Learning

"Fine-tuning" (Transfer Learning) refers to

- adapting a model for a Source Task
- to be able to perform a Target Task

Usually

- this means training a pre-trained model (Source Task)
- with a small number of examples of the Target Task
- causing the model's weights to adapt to the Target Task

But In-Context Learning

- adapts a model for the Source Task
- to be able to solve the Target Task
- **without** changing the model's weights

So this may be an alternative method of Fine-Tuning

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

Done