This note is concerned with getting a language model to do what you want it to do. After the pretraining process, you have a model p_θ parameterized by learned parameters θ which computes the likelihood of all tokens in $w \in \mathcal{V}$ as continuations of a prefix $w_{<t}$, that is, $p_\theta(\cdot \mid w_{<t}) \in \mathbb{R}^{|\mathcal{V}|}$. Now, you have the ambitious goal of constructing a system based on this model that could attempt a broad range of tasks. We'll attempt to cover both the fundamental problems to be solved and core methods.

The language model doesn't know what you want

In this course so far, we've discussed how designing expressive neural networks and training them on large amounts of data can lead to strong **representations**, and a considerable amount of knowledge learned from text. But our interface for interacting with a pretrained language model is, well, a probability distribution. It doesn't know what we want, it just tells us what's likely to come next.

Consider tokenizing the sequence

> What year was Columbia University founded?

with the goal of feeding the tokens to your language model and receiving an answer. Let this be $w_{1:t}$, and consider the distribution $p_\theta(\cdot \mid w_{1:t})$. What are the elements of $\mathcal{V}$ that are likely continuations? Is it 1754, the founding year of Columbia? (Or maybe the tokens 1 or 17 or 175, depending on how the tokenizer tends to tokenize years?) To reason about this, we think, what kind of text was the model trained on? It was trained on a large amount of naturally occuring web text. Surely, the model saw *Columbia University was founded in 1754* at some point, but in this question form it might be more likely that something like the following happens:

> *What year was Columbia University founded?* What year was Princeton University founded? What…

so a likely first new token is *What*. Or it might be likely to generate

> *What year was Columbia University founded?* I've wondered about this for a long time…
> *What year was Columbia University founded?* Great question! In this note, we'll…

The most likely continuation of your sequence isn't necessarily the answer, even if the model **stores knowledge about the answer** in its weights and representations. After all, the model doesn't know what you want, and further questions or follow-up ideas may just be more probable under its estimate of the internet text probability distribution.

Let's look at some methods for making it clearer what we're looking for — leveraging the distribution (and/or parameters) of our pretrained model to make it more useful for our goals.

In-context learning, or, pattern repetition

Consider my question about Columbia University's founding. I can better specify my goals by showing, through the patterns in the input text, the kind of string I want the model to produce. Consider the following updated input:

> *What year was Harvard founded? 1640.*
> *What year was Columbia University founded?*

Before my question about Columbia, I've added what is known as an *in-context example*. It's in-context because this whole prefix we're passing to the model is known as the context, and it's an example of an input (the question) and output (the year of founding) pair that we're looking for. The phrase *in-context learning* refers to the model picking up on the pattern that **likely or probable** continuations to the sequence are years.

Intuitively, by providing in-context examples, we are reducing the *entropy* of the space of outputs, making otherwise-plausible continuations (like following with another related questions) less likely because they don't fit the pattern. This is also often used in, e.g., multiple-choice question answering, to indicate to the model that it should be producing the letter:

Let's set this out in a bit of notation. Consider a dataset of inputs and outputs, $\{(x^{(1)}, y^{(i)})\}_{i=1}^{k}$, and a new input x for which we want to query an answer from our language model. Each input x_i and output y_i is a string over our vocabulary $\mathcal{V}$. If we sample from the language model conditioning on x without using our dataset of examples:

$$\hat{y} \sim p_\theta(y \mid x), \tag{1}$$

we've stated that the problem likely arises that $\hat{y}$ isn't really a good answer to x, but instead just a continuation of x. Recall that this might be done by formatting as:

```
[START] x1 x2 ... xn [SEP]
```

where `[START], [SEP], [END]` are all special tokens in $\mathcal{V}$, and we condition on the sequence up through `[SEP]` and then sample from the model to generate until `[END]`.

Conditioning on some examples from our dataset in-context, also called **few-shot prompting**, is:

$$\hat{y} \sim p_\theta \left(y \mid x, (x^{(1)}, y^{(1)}), \ldots, (x^{(k)}, y^{(k)}) \right), \tag{2}$$

which we might format as:

```
[START] <inp1> x^1_1,... </inp1>
    <out1> y^1_1, ... </out1>
    ...
    <inpk> x^k_1,... </inp1>
    <outk> y^k_1, ... </out1>

    <inpk+1> x1 x2 ... xn </inpk+1>
    <outk+1>
```

So, we've got some extra formatting tags to delimit the start and end of each of the input and the output for the few-shot examples, and we then put the last input—that we're asking our model to answer—and sample a response.

However, one downside of few-shot prompting is that it is computationally expensive at inference time, since the model needs to process long prompts for every input. To address this issue, we can bake in the desired behavior into the model's parameters by adapting them — a process known as **fine-tuning**.