**How to work with large language models**

**How large language models work**

[Large language models](https://openai.com/blog/better-language-models/) are functions that map text to text. Given an input string of text, a large language model predicts the text that should come next.

The magic of large language models is that by being trained to minimize this prediction error over vast quantities of text, the models end up learning concepts useful for these predictions. For example, they learn:

* how to spell
* how grammar works
* how to paraphrase
* how to answer questions
* how to hold a conversation
* how to write in many languages
* how to code
* etc.

None of these capabilities are explicitly programmed in—they all emerge as a result of training.

GPT-3 powers [hundreds of software products](https://openai.com/blog/gpt-3-apps/), including productivity apps, education apps, games, and more.

**How to control a large language model**

Of all the inputs to a large language model, by far the most influential is the text prompt.

Large language models can be prompted to produce output in a few ways:

* **Instruction**: Tell the model what you want
* **Completion**: Induce the model to complete the beginning of what you want
* **Demonstration**: Show the model what you want, with either:
  + A few examples in the prompt
  + Many hundreds or thousands of examples in a fine-tuning training dataset

An example of each is shown below.

**Instruction prompts**

Instruction-following models (e.g., text-davinci-003 or any model beginning with text-) are specially designed to follow instructions. Write your instruction at the top of the prompt (or at the bottom, or both), and the model will do its best to follow the instruction and then stop. Instructions can be detailed, so don't be afraid to write a paragraph explicitly detailing the output you want.

Example instruction prompt:

Extract the name of the author from the quotation below.

“Some humans theorize that intelligent species go extinct before they can expand into outer space. If they're correct, then the hush of the night sky is the silence of the graveyard.”

― Ted Chiang, Exhalation

Output:

Ted Chiang

**Completion prompt example**

Completion-style prompts take advantage of how large language models try to write text they think is mostly likely to come next. To steer the model, try beginning a pattern or sentence that will be completed by the output you want to see. Relative to direct instructions, this mode of steering large language models can take more care and experimentation. In addition, the models won't necessarily know where to stop, so you will often need stop sequences or post-processing to cut off text generated beyond the desired output.

Example completion prompt:

“Some humans theorize that intelligent species go extinct before they can expand into outer space. If they're correct, then the hush of the night sky is the silence of the graveyard.”

― Ted Chiang, Exhalation

The author of this quote is

Output:

Ted Chiang

**Demonstration prompt example (few-shot learning)**

Similar to completion-style prompts, demonstrations can show the model what you want it to do. This approach is sometimes called few-shot learning, as the model learns from a few examples provided in the prompt.

Example demonstration prompt:

Quote:

“When the reasoning mind is forced to confront the impossible again and again, it has no choice but to adapt.”

― N.K. Jemisin, The Fifth Season

Author: N.K. Jemisin

Quote:

“Some humans theorize that intelligent species go extinct before they can expand into outer space. If they're correct, then the hush of the night sky is the silence of the graveyard.”

― Ted Chiang, Exhalation

Author:

Output:

Ted Chiang

**Fine-tuned prompt example**

With enough training examples, you can [fine-tune](https://beta.openai.com/docs/guides/fine-tuning) a custom model. In this case, instructions become unnecessary, as the model can learn the task from the training data provided. However, it can be helpful to include separator sequences (e.g., -> or ### or any string that doesn't commonly appear in your inputs) to tell the model when the prompt has ended and the output should begin. Without separator sequences, there is a risk that the model continues elaborating on the input text rather than starting on the answer you want to see.

Example fine-tuned prompt (for a model that has been custom trained on similar prompt-completion pairs):

“Some humans theorize that intelligent species go extinct before they can expand into outer space. If they're correct, then the hush of the night sky is the silence of the graveyard.”

― Ted Chiang, Exhalation

###

Output:

Ted Chiang

**Code Capabilities**

Large language models aren't only great at text - they can be great at code too. OpenAI's specialized code model is called [Codex](https://openai.com/blog/openai-codex/).

Codex powers [more than 70 products](https://openai.com/blog/codex-apps/), including:

* [GitHub Copilot](https://copilot.github.com/) (autocompletes code in VS Code and other IDEs)
* [Pygma](https://pygma.app/) (turns Figma designs into code)
* [Replit](https://replit.com/) (has an 'Explain code' button and other features)
* [Warp](https://www.warp.dev/) (a smart terminal with AI command search)
* [Machinet](https://machinet.net/) (writes Java unit test templates)

Note that unlike instruction-following text models (e.g., text-davinci-002), Codex is *not* trained to follow instructions. As a result, designing good prompts can take more care.

**More prompt advice**

For more prompt examples, visit [OpenAI Examples](https://beta.openai.com/examples" \o "https://beta.openai.com/examples).

In general, the input prompt is the best lever for improving model outputs. You can try tricks like:

* **Give more explicit instructions.** E.g., if you want the output to be a comma separated list, ask it to return a comma separated list. If you want it to say "I don't know" when it doesn't know the answer, tell it 'Say "I don't know" if you do not know the answer.'
* **Supply better examples.** If you're demonstrating examples in your prompt, make sure that your examples are diverse and high quality.
* **Ask the model to answer as if it was an expert.** Explicitly asking the model to produce high quality output or output as if it was written by an expert can induce the model to give higher quality answers that it thinks an expert would write. E.g., "The following answer is correct, high-quality, and written by an expert."
* **Prompt the model to write down the series of steps explaining its reasoning.** E.g., prepend your answer with something like "[Let's think step by step](https://arxiv.org/pdf/2205.11916v1.pdf)." Prompting the model to give an explanation of its reasoning before its final answer can increase the likelihood that its final answer is consistent and correct.