

upscale, and then put them together in a grid, shown as [Figure 1-14](#). You'll notice that the model doesn't always get the correct number of people (generative AI models are surprisingly bad at math) but it has correctly inferred the general intention by adding more people to the photos on the right than the left.

The output is shown in [Figure 1-14](#).



Figure 1-14. Prompt permutations grid

With models that have APIs like Stable Diffusion, you can more easily manipulate the photos and display them in a grid format for easy scanning. You can also manipulate the random seed of the image, in order to fix a

style in place for maximum reproduceability. With image classifiers it may also be possible to programmatically rate images based on their safe content, or if they contain certain elements associated with success or failure.

5. Divide Labor

As you build out your prompt, you start to get to the point where you're asking a lot in a single call to the AI. When prompts get longer and more convoluted you may find the responses get less deterministic, and hallucinations or anomalies increase. Even if you manage to arrive at a reliable prompt for your task, that task is likely just one of a number of interrelated tasks you need to do your job. It's natural to start exploring how many other of these tasks could be done by AI, and how you might string them together.

One of the core principles of engineering is to use task decomposition to break problems down into their component parts, so you can more easily solve each individual problem and then re-aggregate the results.

Breaking your AI work into multiple calls that are chained together can help you accomplish more complex tasks, as well as provide more visibility into what part of the chain is failing.

There are lots of factors that go into product naming, and an important task is naively outsourced to the AI with no visibility into how it's weighing the importance of these factors (if at all). The way our current system works, we're getting a list of names, but all names are displayed with equal importance with no further context for helping us decide.

Fortunately AI tools are capable of self-evaluation, if we add a second step to our task, we can automatically check for non-desirable outputs:

Input:

Rate this list of product names for a pair of shoes that can fit any foot size. The rating should be out of 10, inline next to the product name:

```
UniFit SoleStrides  
FlexiSize All-Fit Shoes  
OmniFit FootFlex  
...  
AnyStep AdaptiWalkers  
FlexiRange UniversalSteps  
PerfectGlide MultiFit Shoes
```

Output:

```
UniFit SoleStrides: 7/10  
FlexiSize All-Fit Shoes: 8/10  
OmniFit FootFlex: 7/10  
...  
AnyStep AdaptiWalkers: 7/10  
FlexiRange UniversalSteps: 7.5/10  
PerfectGlide MultiFit Shoes: 9/10
```

This ranking is based on clarity, memorability, and how well the name communicates the unique selling point of the product (that they can fit any foot size). Remember, these ratings are subjective and the ultimate success of a product name also depends on the target market, branding, and marketing efforts.

In running this multiple times, it consistently rates the name “OneSize Glovewalkers” as the worst, providing context (if you ask) that the concept might be confusing in a shoe context. You may be wondering why, if the model *knows* this is a bad name, does it suggest it in the first place? LLMs work by predicting the next token in a sequence, and therefore struggle to know what the overall response will be when finished. However, when it has all the tokens from a previous response to review, it can more easily predict whether this would be labeled as a good or bad response.

We can continue to chain multiple calls together to improve the results of our task. For example, we could split this into three separate ratings, on clarity, memorability, and how well the name communicates the unique selling point of the product. These ratings could then be given to a human

as additional context on the final decision, or even calculated together to select the final name programmatically.

The real unlock in learning to work professionally with AI vs just playing around with prompting, is realizing that every part of the system can be broken down into a series of iterative steps. Even with a single prompt this principle applies, as simply appending `Let's think step by step` to the prompt can lead to demonstrable gains in reasoning and proficiency, as well as provide an audit trail for quality assurance and debugging. When taking the time and tokens to reason, the ratings change and are more consistent with the scoring criteria.

Input:

```
Let's think step by step. Rate this list of product names for a pair of shoes that can fit any foot size. The rating should be out of 10, inline next to the product name:
```

```
UniFit SoleStrides
FlexiSize All-Fit Shoes
OmniFit FootFlex
...
AnyStep AdaptiWalkers
FlexiRange UniversalSteps
PerfectGlide MultiFit Shoes
```

Output:

```
Here's a rating out of 10 for each product name, considering factors like descriptiveness, catchiness, and overall appeal:
```

1. **UniFit SoleStrides**: 7/10 - Sounds unifying but may not immediately convey the "fit any size" idea.
 2. **FlexiSize All-Fit Shoes**: 8.5/10 - Combines flexibility and the notion that it fits all sizes.
 3. **OmniFit FootFlex**: 8/10 - Omnipresence (fitting all) and flexibility nicely conveyed.
- ...

18. **AnyStep AdapticWalkers**: 7.5/10 - "AnyStep" suggests versatility, but the overall name might be less direct.
19. **FlexiRange UniversalSteps**: 8/10 - Combines flexibility and universal fit.
20. **PerfectGlide MultiFit Shoes**: 8.5/10 - "MultiFit" is clear, and "PerfectGlide" suggests a comfortable experience.

Remember, the best name can also be influenced by your target audience, branding strategy, and the overall message you want to convey.

OpenAI [calls this](#) “giving the model time to think”, and it is a key tenet of prompt engineering. In effect, *chain of thought* techniques like this, where the model is encouraged to list out its steps, are like dividing a task within the same prompt. Once we’ve automated product naming given a product idea, we can call ChatGPT again to describe each product, which in turn can be fed into Midjourney to generate an image of each product. Using an AI model to generate a prompt for an AI model is *meta prompting*, and it works because LLMs are human-level prompt engineers ([Zhou, 2022](#)).

Input:

Describe the new product "OneFit UltraStride Shoes", a shoe that can fit any foot size, in great detail:

Output:

Introducing the revolutionary "OneFit UltraStride Shoes," the footwear innovation designed to fit any foot size with unparalleled comfort and style. These shoes are a result of cutting-edge technology and meticulous craftsmanship, ensuring a truly personalized and enjoyable walking or running experience for everyone.

Design and Materials:

The OneFit UltraStride Shoes boast an elegant and modern design that appeals to both fashion-forward individuals and sports enthusiasts. They feature a sleek and streamlined silhouette, crafted using premium materials for durability

and aesthetic appeal. The shoe's upper is constructed from a combination of breathable mesh and flexible synthetic materials, allowing for excellent airflow and a lightweight feel.

...

Color and Customization:

OneFit UltraStride Shoes are available in a variety of vibrant colors and stylish patterns, catering to diverse personal preferences. You can choose from classic monochromatic options for a timeless look or opt for bold and vibrant hues to make a fashion statement. The ability to customize the shoe's appearance allows individuals to express their unique style while enjoying the benefits of the OneFit technology.

Conclusion:

The OneFit UltraStride Shoes represent a breakthrough in footwear technology, eliminating the hassle of finding the perfect fit for your feet. With their adaptive design, superior comfort, and stylish aesthetics, these shoes offer a truly personalized experience for individuals of all foot sizes. Stay comfortable, confident, and on-trend with the OneFit UltraStride Shoes, the ultimate choice for those seeking unmatched versatility and a perfect fit.

DALL-E is well-known by GPT-4, and therefore you can invoke its name within ChatGPT and it does a reasonable job at crafting a prompt for an image generation tool. If you were planning on using this prompt in production you may consider applying the prompting principle of providing examples, but it does a good enough job for our purposes without examples.

Input:

Convert this description into a short prompt for an AI image generation model such as DALL-E:

Output:

Create an eye-catching image that visually represents the "OneFit UltraStride Shoes," a revolutionary footwear product that adapts to any foot size. Showcase a sleek and modern shoe design incorporating a stretchable upper, an adjustable lacing system, and a cushioned insole for optimal comfort. Emphasize the shoe's versatility by displaying it in action on various surfaces, highlighting its excellent traction. Incorporate vibrant colors and patterns to convey the customizable nature of the product. The image should capture the essence of innovation, style, and personalized fit that the OneFit UltraStride Shoes offer.

The output of this prompt can now be plugged into image generation tool like DALL-E or Midjourney as a prompt, which can give you a good starting point for visualizing what the product might look like. Although this might not be the final design you go with, seeing an image is more evocative and helps people form an opinion faster. It's easier cognitively to criticize or compliment an existing image than it is to imagine a new image from a blank page or section of text.

The output is shown in [Figure 1-15](#).



Figure 1-15. OneFit UltraStride shoes

It's common practice when working with AI professionally to chain multiple calls to AI together, and even multiple models, in order to accomplish more complex goals. Even single-prompt applications are often built dynamically, based on outside context queried from various databases or other calls to an AI model. The library [LangChain](#) has developed tooling for chaining multiple prompt templates and queries together, making this process more observable and well-structured. A foundational example is progressive summarization, where text that is too large to fit into a context window can be split into multiple chunks of text, with each being summarized, before finally summarizing the summaries. If you talk to builders of early AI products, you'll find they're all under the hood chaining multiple prompts together, called *AI chaining*, to accomplish better results in the final output.

The [Reason and Act \(ReAct\)](#) framework was one of the first popular attempts at AI agents, including the open-source projects [BabyAGI](#), [AgentGPT](#), and [Microsoft AutoGen](#). In effect, these agents are the result of chaining multiple AI calls together in order to plan, observe, act, and then evaluate the results of the action. Autonomous agents will be covered in [Chapter 6](#), but are still not widely used in production at the time of writing. This practice of self-reasoning agents is still early and prone to errors, but there are promising signs this approach can be useful in achieving complex tasks, and is likely to be part of the next stage in evolution for AI systems.

There is an AI battle occurring between large tech firms like Microsoft and Google, as well as a wide array of open-source projects on Hugging Face, and venture-funded startups like OpenAI and Anthropic. As new models continue to proliferate, they're diversifying in order to compete for different segments of the growing market. For example, Anthropic's Claude 2 has an [100,000 token context window](#), compared to GPT-4's standard [8,192 tokens](#). OpenAI soon responded with a [128,000 token window version of GPT-4](#), and Google touts a 1 million token context length with [Gemini 1.5](#). For comparison, one of the Harry Potter books would be around 185,000 tokens, so it may become common for an entire book to fit inside a single prompt, though processing millions of tokens with each API call may be cost-prohibitive for most use cases.

This book focuses on GPT-4 for text generation techniques, as well as Midjourney v6 and Stable Diffusion XL for image generation techniques, but within months these models may no longer be state-of-the-art. This means it will become increasingly important to be able to select the right model for the job, and chain multiple AI systems together. Prompt templates are rarely comparable when transferring to a new model, but the effect of the Five Prompting Principles will consistently improve any prompt you use, for any model, getting you more reliable results.

Summary

In this chapter, you learned about the importance of prompt engineering in the context of generative AI. Prompt engineering is the process of developing effective prompts that yield desired results when interacting with AI models. You discovered that providing clear direction, formatting the output, incorporating examples, establishing an evaluation system, and dividing complex tasks into smaller prompts are key principles of prompt engineering. By applying these principles and using common prompting techniques you can improve the quality and reliability of AI-generated outputs.

You also explored the role of prompt engineering in generating product names and images. You saw how specifying the desired format and providing instructive examples can greatly influence the AI's output.

Additionally, you learned about the concept of role-playing, where you can ask the AI to generate outputs as if it were a famous person like Steve Jobs. The chapter emphasized the need for clear direction and context to achieve desired outcomes when using generative AI models.

Furthermore, you discovered the importance of evaluating the performance of AI models and the various methods used for measuring results, as well as the tradeoffs between quality and token usage, cost, and latency.

In the next chapter, you will be introduced to text generation models. You will learn about the different types of foundation models and their capabilities, as well as their limitations. The chapter will also delve into the standard OpenAI offerings, as well as competitors and open-source alternatives. By the end of the chapter, you will have a solid understanding of the history of text generation models and their relative strengths and weaknesses. The book will return to image generation prompting in Chapters [7](#), [8](#), and [9](#), so you should feel free to skip ahead if that is your immediate need. Get ready to dive deeper into the world of prompt engineering and expand your comfort working with AI.

