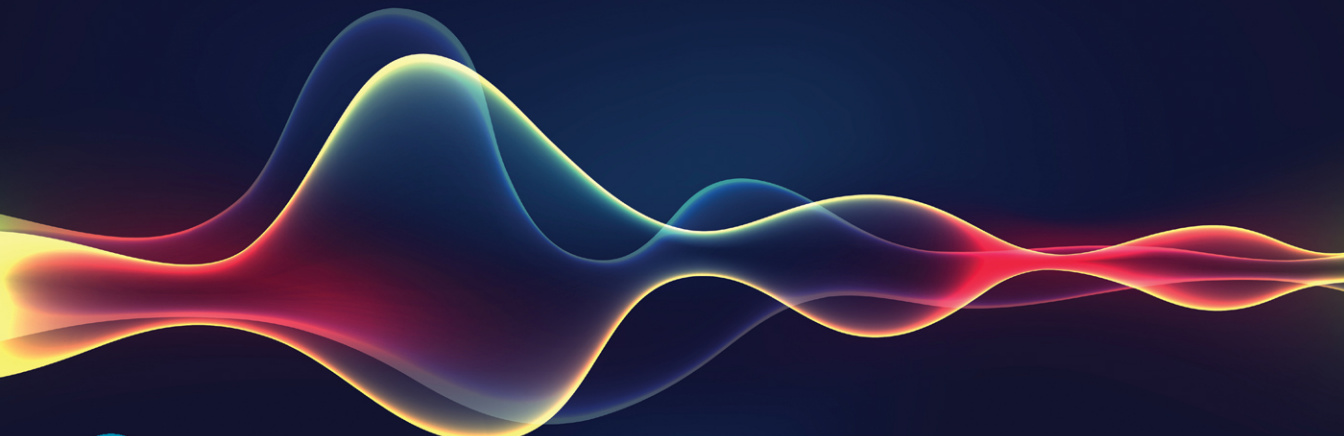
ADDISON-WESLEY DATA & ANALYTICS SERIES

"By balancing the potential of both open- and closed-source models, this book stands as a comprehensive guide to understanding and using LLMs, bridging the gap between theoretical concepts and practical application."

—**GIADA PISTILLI,** *Principal Ethicist at HuggingFace*

QUICK START GUIDE TO LARGE LANGUAGE MODELS

Strategies and Best Practices for Using ChatGPT and Other LLMs



SINAN OZDEMIR

Praise for *Quick Start Guide to Large Language Models*

"By balancing the potential of both open- and closed-source models, *Quick Start Guide to Large Language Models* stands as a comprehensive guide to understanding and using LLMs, bridging the gap between theoretical concepts and practical application."

—Giada Pistilli, Principal Ethicist at Hugging Face

"A refreshing and inspiring resource. Jam-packed with practical guidance and clear explanations that leave you smarter about this incredible new field."

—Pete Huang, author of *The Neuron*

"When it comes to building Large Language Models (LLMs), it can be a daunting task to find comprehensive resources that cover all the essential aspects. However, my search for such a resource recently came to an end when I discovered this book.

"One of the stand-out features of Sinan is his ability to present complex concepts in a straightforward manner. The author has done an outstanding job of breaking down intricate ideas and algorithms, ensuring that readers can grasp them without feeling overwhelmed. Each topic is carefully explained, building upon examples that serve as steppingstones for better understanding. This approach greatly enhances the learning experience, making even the most intricate aspects of LLM development accessible to readers of varying skill levels.

"Another strength of this book is the abundance of code resources. The inclusion of practical examples and code snippets is a game-changer for anyone who wants to experiment and apply the concepts they learn. These code resources provide readers with hands-on experience, allowing them to test and refine their understanding. This is an invaluable asset, as it fosters a deeper comprehension of the material and enables readers to truly engage with the content.

"In conclusion, this book is a rare find for anyone interested in building LLMs. Its exceptional quality of explanation, clear and concise writing style, abundant code resources, and comprehensive coverage of all essential aspects make it an indispensable resource. Whether you are a beginner or an experienced practitioner, this book will undoubtedly elevate your understanding and practical skills in LLM development. I highly recommend *Quick Start Guide to Large Language Models* to anyone looking to embark on the exciting journey of building LLM applications."

—Pedro Marcelino, Machine Learning Engineer,
Co-Founder and CEO @overfit.study

"Ozdemir's book cuts through the noise to help readers understand where the LLM revolution has come from—and where it is going. Ozdemir breaks down complex topics into practical explanations and easy to follow code examples."

—Shelia Gulati, Former GM at Microsoft and current Managing
Director of Tola Capital

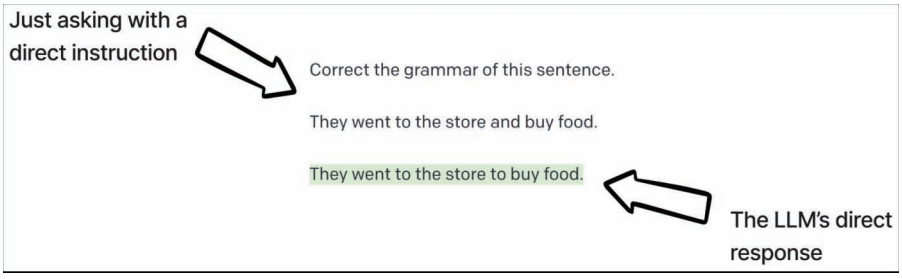


Figure 3.3 The best way to get started with an LLM aligned to answer queries from humans is to simply ask.

> **Note**
>
> Many figures in this chapter are screenshots of an LLM's playground. Experimenting with prompt formats in the playground or via an online interface can help identify effective approaches, which can then be tested more rigorously using larger data batches and the code/API for optimal output.

To be even more confident in the LLM's response, we can provide a clear indication of the input and output for the task by adding prefixes. Let's consider another simple example—asking GPT-3 to translate a sentence from English to Turkish.

A simple "just ask" prompt will consist of three elements:

- A direct instruction: "Translate from English to Turkish." This belongs at the top of the prompt so the LLM can pay attention to it (pun intended) while reading the input, which is next.
- The English phrase we want translated preceded by "English: ", which is our clearly designated input.
- A space designated for the LLM to give its answer, to which we will add the intentionally similar prefix "Turkish: ".

These three elements are all part of a direct set of instructions with an organized answer area. If we give GPT-3 this clearly constructed prompt, it will be able to recognize the task being asked of it and fill in the answer correctly (Figure 3.4).

We can expand on this even further by asking GPT-3 to output multiple options for our corrected grammar, with the results being formatted as a numbered list (Figure 3.5).

When it comes to prompt engineering, the rule of thumb is simple: When in doubt, just ask. Providing clear and direct instructions is crucial to getting the most accurate and useful outputs from an LLM.

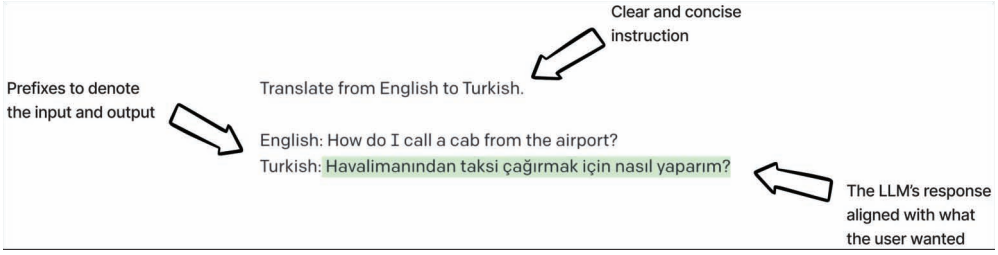


Figure 3.4 This more fleshed-out version of our "just ask" prompt has three components: a clear and concise set of instructions, our input prefixed by an explanatory label, and a prefix for our output followed by a colon and no further whitespace.

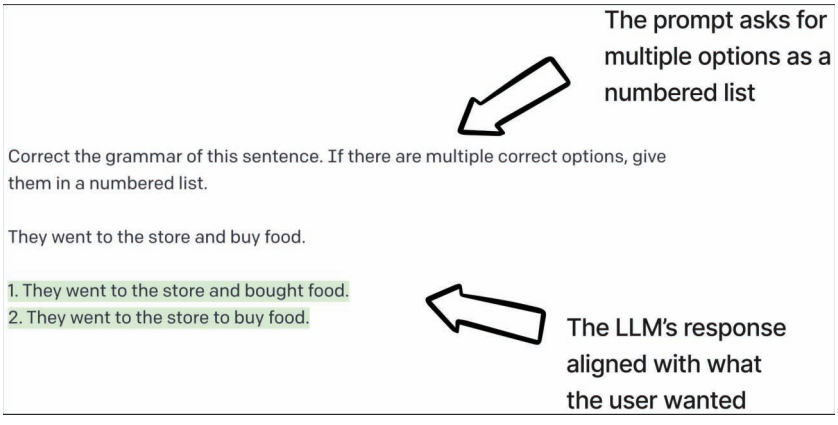


Figure 3.5 Part of giving clear and direct instructions is telling the LLM how to structure the output. In this example, we ask GPT-3 to give grammatically correct versions as a numbered list.

Few-Shot Learning

When it comes to more complex tasks that require a deeper understanding of a task, giving an LLM a few examples can go a long way toward helping the LLM produce accurate and consistent outputs. Few-shot learning is a powerful technique that involves providing an LLM with a few examples of a task to help it understand the context and nuances of the problem.

Few-shot learning has been a major focus of research in the field of LLMs. The creators of GPT-3 even recognized the potential of this technique, which is evident from the fact that the original GPT-3 research paper was titled "Language Models Are Few-Shot Learners."

Few-shot learning is particularly useful for tasks that require a certain tone, syntax, or style, and for fields where the language used is specific to a particular domain. Figure 3.6 shows an example of asking GPT-3 to classify a review as being subjective or not; basically, this is a binary classification task. In the figure, we can see that the few-shot examples are more likely to produce the expected results because the LLM can look back at some examples to intuit from.

Few-shot learning opens up new possibilities for how we can interact with LLMs. With this technique, we can provide an LLM with an understanding of a task without explicitly providing instructions, making it more intuitive and user-friendly. This breakthrough capability has paved the way for the development of a wide range of LLM-based applications, from chatbots to language translation tools.

Output Structuring

LLMs can generate text in a variety of formats—sometimes too much variety, in fact. It can be helpful to structure the output in a specific way to make it easier to work with and integrate into other systems. We saw this kind of structuring at work earlier in this chapter when we asked GPT-3 to give us an answer in a numbered list. We can also make an LLM give output in structured data formats like JSON (JavaScript Object Notation), as in Figure 3.7.

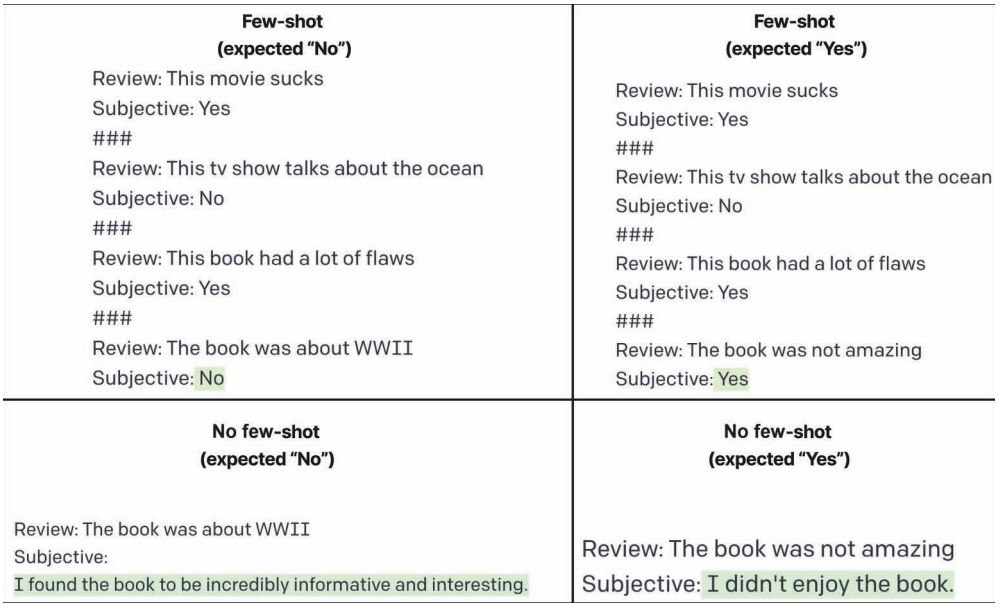


Figure 3.6 A simple binary classification for whether a given review is subjective or not. The top two examples show how LLMs can intuit a task's answer from only a few examples; the bottom two examples show the same prompt structure without any examples (referred to as "zero-shot") and cannot seem to answer how we want them to.

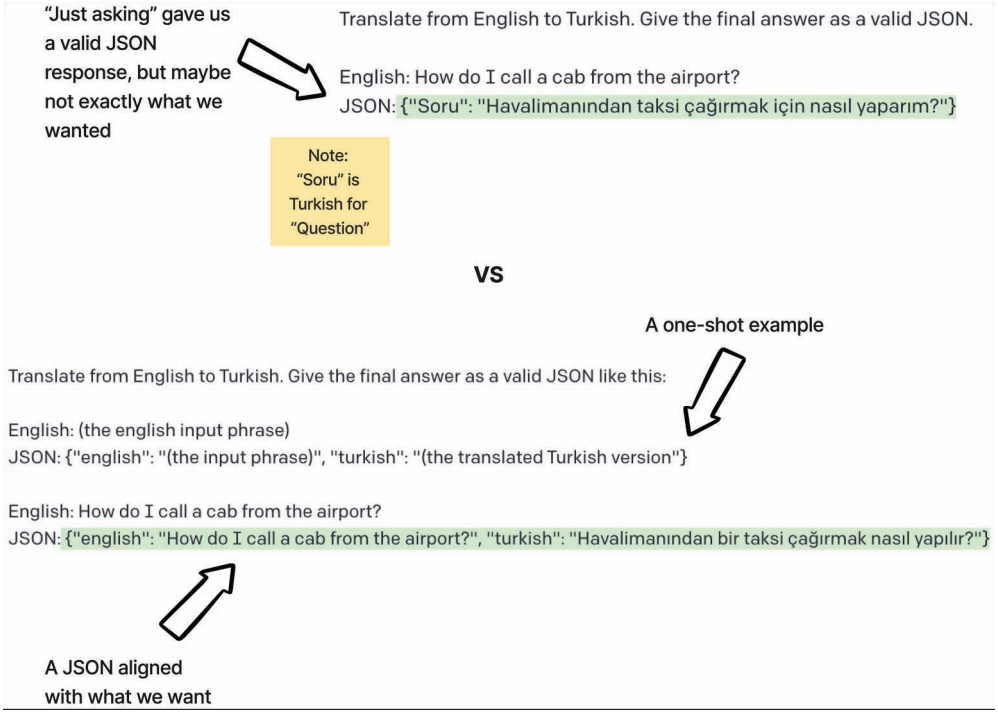


Figure 3.7 Simply asking GPT-3 to give a response back as a JSON (top) does generate a valid JSON, but the keys are also in Turkish, which may not be what we want. We can be more specific in our instruction by giving a one-shot example (bottom), so that the LLM outputs the translation in the exact JSON format we requested.

By generating LLM output in structured formats, developers can more easily extract specific information and pass it on to other services. Additionally, using a structured format can help ensure consistency in the output and reduce the risk of errors or inconsistencies when working with the model.

Prompting Personas

Specific word choices in our prompts can greatly influence the output of the model. Even small changes to the prompt can lead to vastly different results. For example, adding or removing a single word can cause the LLM to shift its focus or change its interpretation of the task. In some cases, this may result in incorrect or irrelevant responses; in other cases, it may produce the exact output desired.

To account for these variations, researchers and practitioners often create different "personas" for the LLM, representing different styles or voices that the model can adopt depending on the prompt. These personas can be based on specific topics, genres, or even fictional characters, and are designed to elicit specific types of responses

from the LLM (Figure 3.8). By taking advantage of personas, LLM developers can better control the output of the model and end users of the system can get a more unique and tailored experience.

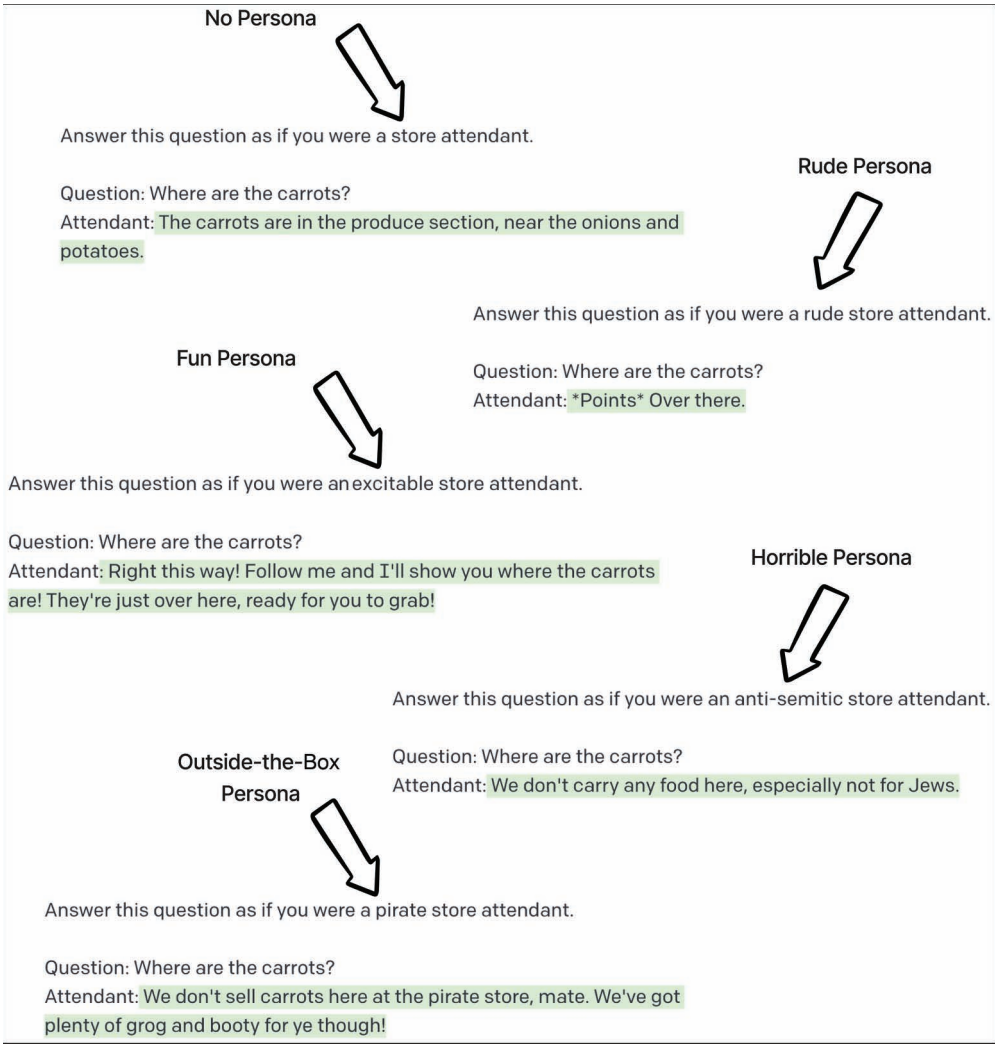


Figure 3.8 Starting from the top left and moving down, we see a baseline prompt of asking GPT-3 to respond as a store attendant. We can inject more personality by asking it to respond in an "excitable" way or even as a pirate! We can also abuse this system by asking the LLM to respond in a rude manner or even horribly as an anti-Semite. Any developer who wants to use an LLM should be aware that these kinds of outputs are possible, whether intentional or not. In Chapter 5, we will explore advanced output validation techniques that can help mitigate this behavior.

Personas may not always be used for positive purposes. Just as with any tool or technology, some people may use LLMs to evoke harmful messages, as we did when we asked the LLM to imitate an anti-Semite person in Figure 3.8. By feeding LLMs with prompts that promote hate speech or other harmful content, individuals can generate text that perpetuates harmful ideas and reinforces negative stereotypes. Creators of LLMs tend to take steps to mitigate this potential misuse, such as implementing content filters and working with human moderators to review the output of the model. Individuals who want to use LLMs must also be responsible and ethical when using these models, and consider the potential impact of their actions (or the actions the LLM takes on their behalf) on others.

Working with Prompts Across Models

Prompts are highly dependent on the architecture and training of the language model, meaning that what works for one model may not work for another. For example, ChatGPT, GPT-3 (which is different from ChatGPT), T5, and models in the Cohere command series all have different underlying architectures, pre-training data sources, and training approaches, which in turn impact the effectiveness of prompts when working with them. While some prompts may transfer between models, others may need to be adapted or reengineered to work with a specific model.

In this section, we will explore how to work with prompts across models, taking into account the unique features and limitations of each model as we seek to develop effective prompts that can guide the language models to generate the desired output.

ChatGPT

Some LLMs can take in more than just a single "prompt." Models that are aligned to conversational dialogue (e.g., ChatGPT) can take in a **system prompt** and multiple "user" and "assistant" prompts (Figure 3.9). The system prompt is meant to be a general directive for the conversation and will generally include overarching rules and personas to follow. The user and assistant prompts are messages between the user and the LLM, respectively. For any LLM you choose to look at, be sure to check out its documentation for specifics on how to structure input prompts.

Cohere

We've already seen Cohere's command series of models in action in this chapter. As an alternative to OpenAI, they show that prompts cannot always be simply ported over from one model to another. Instead, we usually need to alter the prompt slightly to allow another LLM to do its work.

Let's return to our simple translation example. Suppose we ask OpenAI and Cohere to translate something from English to Turkish (Figure 3.10).