Prompts: Your Secret Weapon

Prompts are your passport to effective communication with language models. Think of them as the starting point of a conversation or the first step in giving directions. When you provide a prompt for a language model, you're setting the stage for what comes next — the model's response.

Prompts can be as simple as "Translate the following English text to French" or as complex as a string of sentences providing context for a question. **Essentially, prompts can range from single-word commands to multi-paragraph discourses.**

In this unit, you will experiment with prompts while learning to improve on them and analyze how different prompts yield different results! Please create a free Chat-GPT account now if you have not already done so.

The Power of Prompts

Prompts play a vital role in controlling a language model's behavior. The choice of prompt can vastly change the output from the model. Asking GPT-3 to "Write a paragraph about elephants" yields a general description of elephants. However, if you prompt it with "Write a paragraph about the role of elephants in ecosystem preservation," the response will be more specific and focused on the topic at hand.

A well-crafted prompt can make the difference between getting a generic response and a highly specific, useful answer. The power of prompts shines through when you provide control and specificity when interacting with language models.

Craft Your Prompts

Now that you understand what prompts are and why they're so powerful, let's learn how to craft them effectively.

Be Explicit	Language models, as intelligent as they may be, don't understand subtlety or implied meanings as humans do. Therefore, it's essential to be clear and explicit in your prompts. For example, if you're seeking a summary of a text, state it clearly: "Provide a summary of the following text:"
Context is Key	Language models generate responses based on the given prompt and their trained context. To steer the model's output in a certain direction, consider providing context in the prompt. For example, "As a professional chef, how would you prepare a steak?" will likely yield a different response than "As a vegan chef, how would you prepare a steak?"
Experiment	Remember, there's no one-size-fits-all approach to crafting prompts. What works best often depends on the specific task and the behavior of the language model. Don't hesitate to iterate and experiment with different prompt structures to achieve your desired output.

To help you practice crafting prompts, let's do a quick exercise. Consider the following scenario:

You're using a language model to generate ideas for a story set in a post-apocalyptic world. What would be your prompt? Write down several versions of prompts on the Chat-GPT chatbot window and compare the responses!

Remember, there's no 'right' answer here! The goal is to generate a compelling and context-rich response from the language model.

A Look Under the Hood of Generative AI

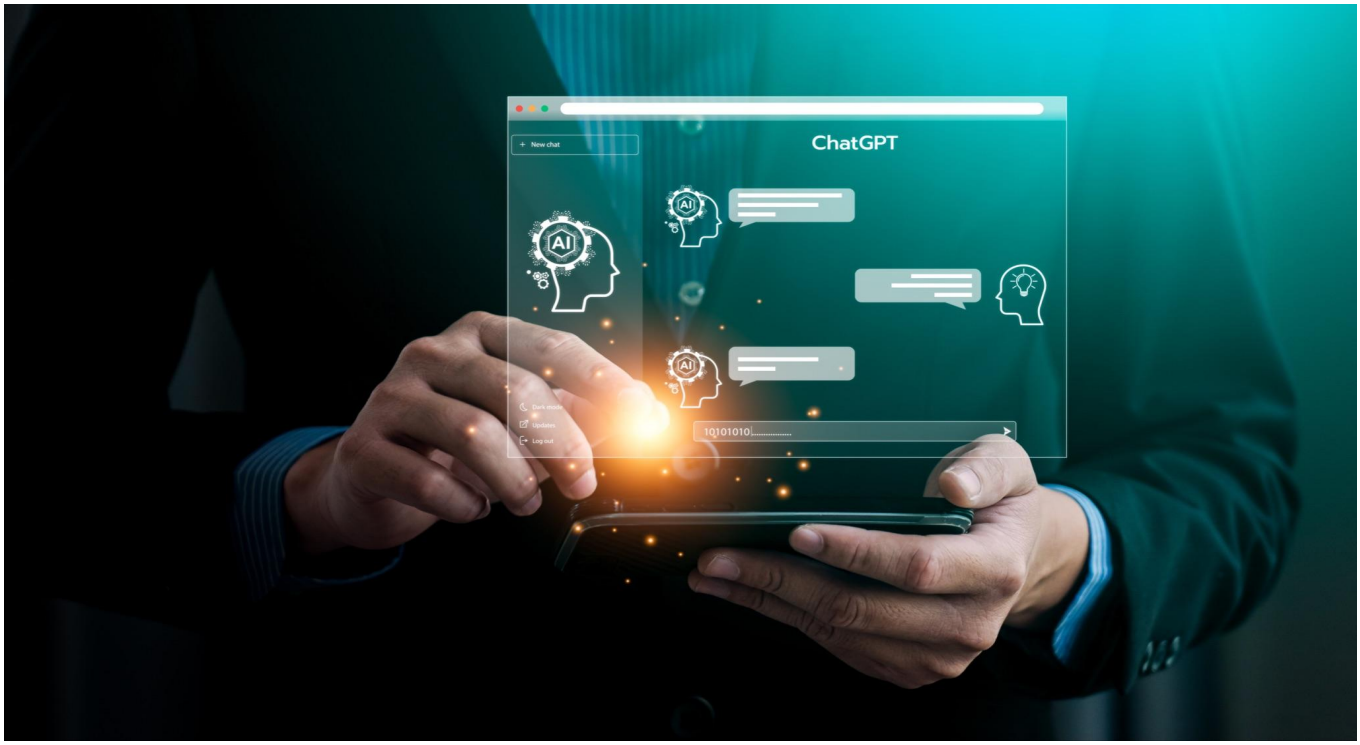
Now that you have a basic understanding of prompts, let's take a peek under the hood and understand how prompts work within the framework of generative AI.

Generative AI models, like GPT-3 and GPT-4, learn to generate outputs by being trained on a vast amount of text data. They learn the statistical properties of the language, such as common sequences of words, sentence structures, and topic

dependencies. During this training, these models learn to predict the next word in a sentence based on all the previous words.

They are called '**transformer-based language models**' because they transform an input sequence of words (a prompt) into an output sequence of words (a response).

The Prompt-Response Mechanism



Let's look at the fascinating journey of a prompt inside a generative model. **When you feed a prompt to a model like GPT-3 or GPT-4, the model processes it word by word.** The model updates its understanding of the context with each word and prepares to generate the next word.

Once the prompt is entirely processed, the model begins generating the response, one word at a time. It chooses each word based on the context provided by the prompt and all the previously generated words in the response.

For example, If you give the prompt, "Once upon a time in a kingdom far away," the model, based on its training, recognizes this as the beginning of a story and will likely generate a response that continues this narrative in a story-like manner.

The Magic of Context

Prompts do more than just initiate a conversation. They set the tone, guide the content, and even influence the style of the response. The more context a prompt provides, the better the model can align its response to the desired outcome.

For example, if you ask the model, "Who won the world series in 2020?" without any additional context, the model will give you the correct answer based on its training data. But if you provide additional context like "As a sports commentator, describe the final moments when the team won the world series in 2020," the model generates a much more descriptive and immersive response like a sports commentator would.

Fine-Tuning With Prompts

Apart from steering the immediate response of a model, prompts can also be used to 'fine-tune' the model's behavior. Fine-tuning is a process of additional, targeted training on top of the base model.

During fine-tuning, the model undergoes training on a dataset created with carefully designed prompts and responses. Fine-tuning aims to make the model more useful, ensure its safety, or make it an expert in a specific domain.

For example, to fine-tune a model to generate Python code, you might create a dataset where each prompt is a plain-English description of a programming task, and each response is a Python code snippet that accomplishes the task.

The Unseen Workhorse of Language Models: Tokenizers

Before we talk more about prompts, let's take a moment to appreciate an essential yet often overlooked component of language models: the tokenizer. **The tokenizer's job is to break down the input text (our prompt) into smaller parts, or 'tokens,' which the model can understand.**

In English, this can be as simple as breaking sentences down into individual words. But for GPT-3 and GPT-4, which can understand multiple languages and even generate code, the tokenizer has a more complicated task. **It uses a method called 'byte-pair encoding' to break down text into tokens that can represent words, parts of words, or even individual characters.**

For instance, the sentence "ChatGPT is fun to interact with" could be broken down into tokens like ["Chat", "G", "PT", " is", " fun", " to", " interact", " with"]. Notice how spaces before words are part of the tokens — this helps the model keep track of word boundaries. Understanding tokenization is crucial as it impacts the way you craft your prompts.

The Transformer Architecture and Prompts

Now that we've tokenized our prompt, it's time to feed these tokens into the model. Here's where the 'transformer' part of 'transformer-based language models' comes in.

The transformer architecture is the fundamental building block of modern language models like GPT-3 and GPT-4. It was introduced in a paper titled "Attention is All You Need" by Vaswani et al. (2017), and it has been a game-changer in the field of Natural Language Processing (NLP).

The transformer model processes all tokens simultaneously within a prompt, unlike previous models that handled tokens sequentially, one at a time. This parallel processing is one of the reasons why transformer models are so powerful and efficient.

Inside the transformer model, each token is transformed into a high-dimensional vector using a process called embedding. These vectors capture the semantic meaning of the tokens.

Then, the model calculates 'attention scores,' which determine how much each token in the prompt should contribute to understanding every other token. This is the **'self-attention' mechanism**, and it allows the model to capture the dependencies among all tokens, no matter how far apart they are in the prompt.

The Decoding Process: From Tokens to Text

Once the model has processed the prompt, it starts generating the response, one token at a time. For each token, the model calculates a probability distribution over the entire vocabulary, which can be tens of thousands of tokens, and selects the token with the highest probability as the next token in the response.

This is where the temperature parameter comes in. A higher temperature makes the model more 'creative' by increasing the randomness in the token selection process, while a lower temperature makes the model more 'deterministic' by sticking closely to the most probable tokens.

Once the model has decided on the next token, it adds this token to the context (i.e., the prompt and the previously generated tokens) and repeats the process to generate the next token, and so on, until it generates the end-of-text token or reaches the maximum token limit.

The following is a step-by-step process of the entire sequence of events from when a user interacts with the LLM to when they receive a response:

1. **Input Text**: The process begins with the input text, which could be a prompt or a response.
2. **Tokenization**: Tokenization is the process of breaking down the input text into individual tokens. Tokens can be words, characters, or even smaller units, depending on the specific tokenization strategy.
3. **Special Tokens:** In prompt engineering, special tokens are often used to provide additional instructions or guidance to the language model. These tokens may include prompts like "Translate the following sentence into French:" or "Summarize the given article: ."Special tokens help in framing the context and purpose of the prompt.
4. **Token Encoding**: Each token is encoded into a numerical representation that the language model can process. This encoding is typically achieved using techniques like one-hot encoding, word embeddings, or subword embeddings.
5. **Token Sequence:** The individual tokens, along with their encoded representations, form a token sequence. This sequence preserves the order of the tokens in the original text.
6. **Padding and Truncation:** In cases where the token sequence exceeds a predefined length limit, padding or truncation techniques may be applied to ensure all sequences have a consistent length. Padding involves adding special tokens to make shorter sequences equal in length, while truncation involves removing tokens from longer sequences.
7. **Model Input**: The token sequence is then used as input to the language model for generating the desired output or response.
8. **Model Processing:** The language model processes the token sequence and generates a response based on the provided prompt and its internal understanding of language patterns.
9. **Decoding:** The generated response from the language model is decoded to obtain the final output, which can be text, translations, summaries, or other desired outcomes.

It's important to note that tokenization techniques may vary based on the specific language model used and the requirements of the prompt engineering task.