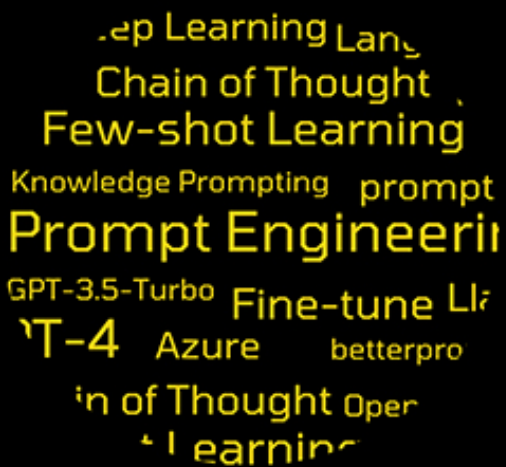


LLM Prompt Engineering For Developers

The Art and Science of Unlocking LLMs' True Potential

Aymen EL Amri

www.faun.dev

LLM Prompt Engineering For Developers

The Art and Science of Unlocking LLMs' True Potential

Aymen El Amri

www.faun.dev

LLM Prompt Engineering For Developers

The Art and Science of Unlocking LLMs' True Potential

Published by FAUN.

For permission requests, write to the author at the provided email address: aymen@faun.dev

work are the property of their respective owners. FAUN is not associated with any product or vendor mentioned in this book.

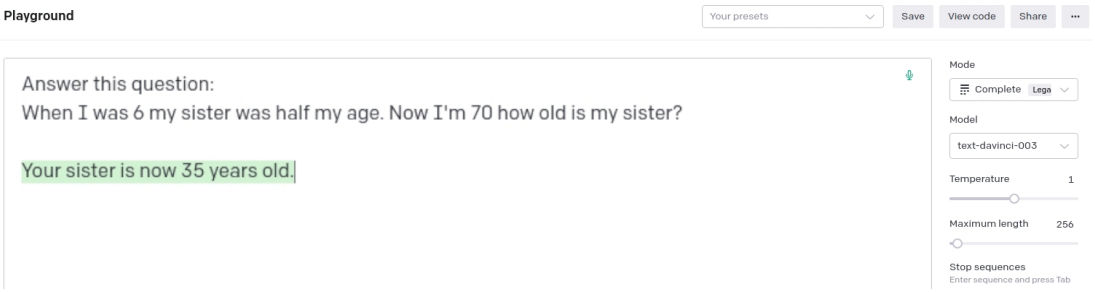
For general information on our other products and services or to obtain technical support, please contact us at sales@faun.dev

FAUN is a registered trademark of eralabs SASU, a French company.

Table of Contents

1 - Preface

Many of you have likely experimented with ChatGPT or other Large Language Models (LLMs) and initially been swept up in the excitement. The initial “wow” effect was undeniable. However, after a few interactions, the shine began to wear off. You might have noticed its quirks and inconsistencies, realizing that, in its raw form, it doesn’t seamlessly fit into our daily tasks.



text-davinci-003 fail

Have you ever asked a language model to write a blog post on a specific topic, only to receive a response that was off-mark or lacked depth? Or perhaps you asked it to summarize a long and complex article, and instead of a concise summary, you got a jumbled mix of sentences lacking the main points. Such experiences can be frustrating, especially when you know the potential that lies within these models.

You may have searched for “perfect” prompts in an online prompt database - a new phenomenon that has emerged in the wake of ChatGPT’s and other LLMs’ popularity - hoping for a quick solution. However, you soon realize that what works for one person may not work for another.

Each prompt is unique and tailored to specific needs and contexts. It is a mistake to assume that the one-size-fits-all approach will always work. In all cases, be assured: This guide goes beyond just providing prompts and is not a collection of prompts. Instead, it explains the “how” behind crafting

effective prompts. With this knowledge and skill, you can create prompts that align with your objectives and lead to better interactions with LLMs.

In reality, if you were sometimes disappointed, it's not entirely the LLM's fault. The key lies in how we communicate with it. Just as we need to frame our questions correctly when seeking answers from a search engine, we need to craft our prompts effectively for ChatGPT or any other LLM for that matter. It's just that search engines do not need as much guidance as LLMs do. This art of guiding LLMs and optimizing our interactions with them is more than just a technique; it's a paradigm shift, a new experimental science that we call "Prompt Engineering".

Indeed, it underscores the importance of clear, structured communication with AI. As developers, we're not merely coding; we're teaching, guiding, and refining the AI's understanding. This guide, "LLM Prompt Engineering For Developers", serves as your comprehensive guide to mastering this emerging discipline. Dive in and discover how to unlock the true potential of ChatGPT and similar models, transforming them into invaluable tools in your development arsenal.

The art of communicating with LLMs heavily relies on prompt engineering. Consider it as finding the perfect key for a lock. While humans can interpret vague or incomplete information, LLMs require precise instructions to deliver optimal results. The goal isn't to converse in human terms, but to effectively translate our needs into a language the model comprehends.

Often, individuals approach LLMs as if they're interacting with another human, projecting their own thought processes and expecting similar interpretations. However, this anthropomorphic perspective can be misleading. Instead of expecting the LLM to think like us, we must strive to understand its "logic" so that it can better serve our needs. This mutual understanding forms the essence of prompt engineering. By refining our prompts, we guide the LLM to produce the most relevant and insightful responses.

Prompt engineering is not an isolated discipline; it stands at the intersection of several fields, each contributing unique insights. From Artificial Intelligence algorithms to data-driven strategies in Data Science and

predictive models in Machine Learning, prompt engineering draws from a rich tapestry of knowledge. This multidisciplinary nature might seem daunting initially, suggesting a steep learning curve that requires expertise in various domains.

To excel in this field, possessing robust skills in Artificial Intelligence, Data Science, and Machine Learning is essential. Proficiency in these areas is invaluable for understanding the workings of generative AIs and writing prompts with greater precision and accuracy.

However, this guide, “LLM Prompt Engineering For Developers,” aims to demystify the complexity of prompt engineering. Its goal is to distill the essence of prompt engineering into its most accessible form. While a foundational understanding of these disciplines certainly aids in mastering prompt engineering, it’s not a strict prerequisite to start practicing and seeing tangible results.

“LLM Prompt Engineering For Developers” is designed to be a bridge, allowing anyone with a basic grasp of programming to dive into the world of prompt engineering and prompt engineering agility. Through clear explanations, practical examples, and step-by-step hands-on labs, we’ll simplify the journey, ensuring that you not only understand the core concepts but also feel empowered to take your first steps in this exciting field.

1.1 - What Are You Going to Learn?

In “LLM Prompt Engineering For Developers,” we take a comprehensive journey into the world of LLMs and the art of crafting effective prompts for them.

The guide starts by laying the foundation, exploring the evolution of Natural Language Processing (NLP) from its early days to the sophisticated LLMs we interact with today. You will dive deep into the complexities of models such as GPT models, understanding their architecture, capabilities, and nuances.

As we progress, this guide emphasizes the importance of effective prompt engineering and its best practices. While LLMs like ChatGPT (GPT-3.5) are powerful, their full potential is only realized when they are communicated with effectively. This is where prompt engineering comes into play. It's not simply about asking the model a question; it's about phrasing, context, and understanding the model's logic.

Through chapters dedicated to Azure Prompt Flow, LangChain, and other tools, you'll gain hands-on experience in crafting, testing, scoring, and optimizing prompts. We'll also explore advanced concepts like Few-shot Learning, Chain of Thought, Perplexity, and techniques like ReAct and General Knowledge Prompting, equipping you with a comprehensive understanding of the domain.

This guide is designed to be hands-on, offering practical insights and exercises. In fact, as you progress, you'll familiarize yourself with several tools:

- **OpenAI Python library**: You will dive into the core of OpenAI's LLMs and learn how to interact and fine-tune models to achieve precise outputs tailored to specific needs.
- **Promptfoo**: You will master the art of crafting effective prompts. Throughout the guide, we'll use Promptfoo to test and score prompts, ensuring they're optimized for desired outcomes.
- **LangChain**: You'll explore the LangChain framework, which elevates LLM-powered applications. You'll dive into understanding how a prompt engineer can leverage the power of this tool to test and build effective prompts.
- **Betterprompt**: Before deploying, it's essential to test. With Betterprompt, you'll ensure the LLM prompts are ready for real-world scenarios, refining them as needed.
- **Azure Prompt Flow**: You will experience the visual interface of Azure's tool, streamlining LLM-based AI development. You'll design executable flows, integrating LLMs, prompts, and Python tools, ensuring a holistic understanding of the art of prompting.
- And more!

With these tools in your toolkit, you will be well-prepared to craft powerful and effective prompts. The hands-on exercises will help solidify your understanding. Throughout the process, you'll be actively engaged and by the end, not only will you appreciate the power of prompt engineering, but you'll also possess the skills to implement it effectively.

1.2 - To Whom is This Guide For?

This guide is designed for those passionate about LLMs and the emerging field of prompt engineering. Regardless of your AI expertise or familiarity with programming, this guide provides a clear route to mastering the nuances of creating effective prompts for LLMs. So if you are a beginner, don't worry. We'll start from the basics and build up from there. For those eager to unlock the vast capabilities of language models in practical scenarios, consider this guide your essential starting point.

1.3 - Join the Community

This guide was published by FAUN, a community of developers, architects, and software engineers who are passionate about learning and sharing their knowledge. If you're interested in joining the community, you can start by subscribing to our newsletter at faun.dev/join. Every week, we share the most important and relevant articles, tutorials, and videos on the latest technologies and trends, including AI, ML, and NLP. You can also follow us on Twitter at @joinFAUN and LinkedIn to stay up-to-date with the latest news and announcements.

1.4 - About the Author

Aymen El Amri is a software engineer, author, and entrepreneur. He is the founder of the FAUN Developer Community. He is also the author of multiple books on software engineering. You can find him on Twitter and LinkedIn.

1.5 - The Companion Toolkit

For an enhanced learning experience, we have created a companion toolkit that includes all the code snippets and prompts used in this guide. You can download it using the following URL: https://from.faun.to/r/llmcd or by scanning the QR code below.

QR Code

1.6 - Your Feedback Matters

Countless hours have been poured into crafting this guide, aiming to provide you with the best knowledge and tools to master prompt engineering. I would love to hear your thoughts and suggestions. Your feedback will help me improve this guide and create better content in the future. Your insights could also light the way for others! If you feel I've achieved my goal and this guide has been valuable to you, I'd be deeply honored if you shared your experience with a thoughtful review on the marketplace where you purchased this book, on social media or directly with me via email (aymen@faun.dev).

Your feedback will help me quantify the impact of this work and help others discover this book and benefit from the knowledge within.

2 - From NLP to Large Language Models

2.1 - What is Natural Language Processing?

Natural language refers to the language that humans use to communicate with each other. It encompasses spoken and written language, as well as sign language. Natural language is distinct from formal language, which is used in mathematics and computer programming.

Generative AI systems, specifically ChatGPT, are capable of understanding and producing both natural and formal languages. In both cases, interactive AI assistants like ChatGPT use natural language to communicate with humans. Their output could be a natural language response or a mix of natural and formal languages.

To process, understand, and generate natural language, a whole field of AI has emerged: Natural Language Processing (NLP). NLP, by definition, is the field of artificial intelligence that focuses on the understanding and generation of human language by computers. It is employed in a wide range of applications, including voice assistants, machine translation, chatbots, and more. In other words, when we talk about NLP, we refer to the ability of computers to understand and generate natural language.

NLP has experienced rapid growth in recent years, largely due to advancements in language models such as GPT and BERT. These models are some of the most powerful NLP models to date. But what is a language model?

2.2 - Language Models

Models are intelligent computer programs that can perform a specific task. For example, a model can be trained to recognize images of cats and dogs,

to write social media posts or blog posts, to provide medical assistance or legal advice, and so on.

These models are the result of a training process that uses large datasets to teach the model how to perform a specific task. The larger the dataset, the more accurate the model will be. This is why models trained on large datasets are often more accurate than models trained on smaller datasets.

Using the dataset used for training, models acquire the capability to make predictions on new data. For example, a model trained on a dataset of images of cats and dogs can predict whether a new image contains a cat or a dog.

Language models are a subset of models capable of generating, understanding, or manipulating text or speech in natural language. These models are essential in the field of NLP and are used in various applications such as machine translation, speech recognition, text generation, chatbots, and more.

Here are some types of language models:

- Statistical models (n-grams)
- Neural network-based models
    - Feedforward neural networks
    - Recurrent neural networks (RNNs)
    - Long short-term memory (LSTM)
    - Gated recurrent units (GRUs)
- Knowledge-based models
- Contextual language models
- Transformer models
    - Bidirectional encoder representations from transformers (BERT)
    - Generative pre-trained transformer (GPT)

2.3 - Statistical Models (N-Grams)

Statistical models, like n-gram models, serve as foundational language models commonly used for text classification and language modeling. They

can also be adapted for text generation, although more advanced models are typically better suited for complex text-to-text tasks. Within statistical models, word sequence probabilities are derived from training data, enabling the model to estimate the likelihood of the next word in a sequence.

N-gram models specifically consider the preceding n-1 words when estimating the probability of the next word. For instance, a bigram model takes into account only the preceding single word, while a trigram model examines the two preceding words. This characteristic endows n-gram models with quick training and utilization capabilities, but they exhibit limitations in capturing long-range dependencies.

ℹ️ In a trigram model, each current word is paired with the two preceding words, forming sequences of three words. For instance, in the sentence “A man of knowledge restrains his words,” the observed trigrams would include “A man of,” “man of knowledge,” “of knowledge restrains,” “knowledge restrains his,” and “restrains his words.” These sequential 3-word patterns are then employed by the model to estimate the probabilities of subsequent words.

Rather than clustering words, n-gram models leverage local word order and context derived from the training data. By focusing on these short-term sequences, n-gram models can make predictions about forthcoming words without modeling global semantics. Although they are efficient and straightforward, their local context makes them less suitable for generating lengthy texts.

Statistical models, particularly n-grams, are quite different from the more recent neural language models. The concept of “prompt engineering” as it’s understood today is more closely associated with the latter. However, there are ways in which the design of input or the preprocessing of data for n-gram models can be thought of as a precursor to prompt engineering.

2.4 - Knowledge-Based Models

These models combine NLP techniques with a structured knowledge base, enabling them to perform tasks that require a deeper understanding and reasoning. They are more useful in specific domains such as medicine or law.

2.5 - Contextual Language Models

These models can understand the meaning of words based on their context. ELMo (Embeddings from Language MOdels) is an example of a contextual language model. ELMo is primarily used to obtain word representations that take context into account. These representations can then be used in various NLP tasks such as text classification, named entity recognition, and more.

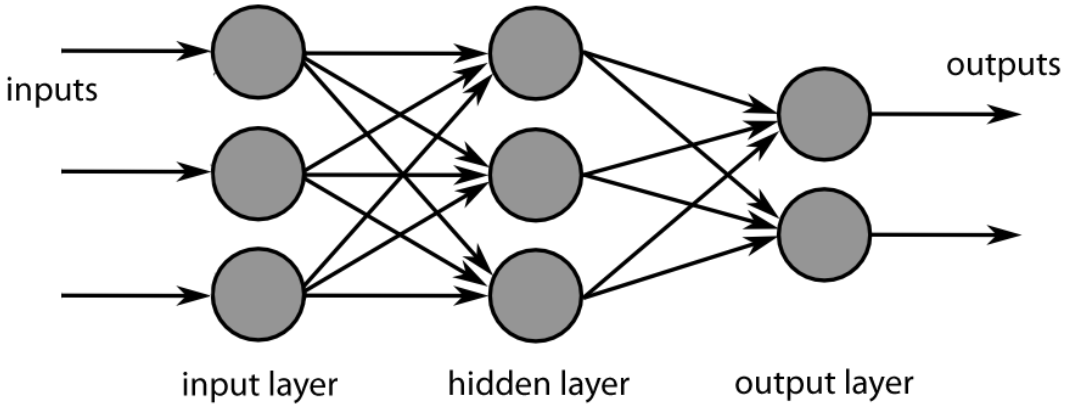
2.6 - Neural Network-Based Models

Neural network-based models are models that learn and process information in a way inspired by the human brain. They are designed to recognize patterns and make predictions based on large amounts of data. These models consist of interconnected artificial neurons that work together to solve tasks such as image recognition, natural language processing, voice recognition, and more!

Neural network-based models are used in various applications, including self-driving cars, virtual assistants, and recommendation systems. They enable computers to learn from examples and improve their performance over time.

2.6.1 - Feedforward Neural Networks

Feedforward neural networks are the simplest type of neural network. They consist of an input layer, one or more hidden layers, and an output layer. The input layer receives the input data, which is then passed through the hidden layers to the output layer.



Multilayer Neural Network

Each layer consists of a set of neurons that perform a specific task. The neurons in the input layer receive the input data and pass it to the neurons in the hidden layers. The neurons in the hidden layers perform calculations on the input data and pass the results to the neurons in the output layer. The neurons in the output layer perform calculations on the results and produce the final output.

This type of neural network can be used for tasks such as image recognition, speech recognition, and other simple classification tasks. However, it is not suitable for more complex tasks.

2.6.2 - Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) were created to overcome the limitations of traditional feedforward neural networks in handling sequential data. Unlike feedforward networks, which process inputs independently, RNNs have the ability to retain information from previous steps in the sequence. This is what makes them well-suited for handling sequential data like text and speech.

They are used in various applications such as translation, sentiment analysis, and text-to-speech processing.

ℹ️ Sequential data is data that is ordered in a particular way, and each element in the sequence has a specific meaning. For example, a sentence is

a sequence of words, and each word has a specific meaning.

2.6.3 - Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a special type of neural network that excels at understanding and remembering information in sequences of data. It was created to solve a problem that regular neural networks have with remembering things over long periods.

Regular neural networks can sometimes forget important information when dealing with sequences. But LSTM is designed to remember important details and pass them along through many steps in the sequence.

This makes LSTM useful for tasks where understanding the order and context of the data is important, such as language translation, speech recognition, and predicting the next word in a sentence.

ℹ️ LSTM is like a smart memory that helps the neural network remember things in the right order and context.

They can be used in text generation tasks. While you can provide an LSTM with an initial sequence (akin to a “prompt”) to generate or classify subsequent sequences, the nuanced manipulation of this initial sequence to guide the LSTM’s output is not typically referred to as “prompt engineering.”

2.6.4 - Gated Recurrent Units (Grus)

Gated Recurrent Units (GRUs) are a type of neural network architecture designed to process sequences of data, much like Long Short-Term Memory (LSTM) networks. One of the primary motivations behind the development of GRUs was to address some of the complexities and computational demands of LSTMs, while still effectively capturing long-term dependencies in sequential data. As a result, GRUs have a more streamlined structure than LSTMs, which often allows them to train faster and require fewer computational resources. A key feature of GRUs is the use of “gates.”

ℹ️ Think of gates as checkpoints that regulate the flow of information within the network. They determine what data should be retained, updated, or discarded as the sequence is processed. This mechanism ensures that the network focuses on relevant details and can recall important information from earlier in the sequence.

GRUs have proven valuable in a variety of tasks that require understanding sequences, such as language processing, speech recognition, and more. Their ability to recognize patterns and relationships in sequential data makes them a popular choice for many applications in the realm of deep learning.

Text generation is one such application, but the manipulation of the initial sequence to guide the GRU's output is not typically referred to as "prompt engineering."

2.7 - Transformer Models

Imagine you're reading a book, word by word. You understand each word, but you don't understand the meaning of the whole sentence. You need to read the whole sentence to understand it. Similarly, you need to read the whole paragraph to understand the meaning of the whole paragraph. And you need to read the whole book to understand the meaning of the whole book. Naturally, this is how humans read and understand a book. The basic idea here is that you need to read the whole text to understand it, and it all starts with understanding each word and how it relates to the other words around it.

The transformer operates in a similar way. Instead of reading word by word, it can examine multiple words at once and understand their relationships. This is known as "attention".

A few years ago, some researchers at Google devised a new method for reading text. They named it the "transformer", and it was described in a paper titled '[Attention Is All You Need]' by Ashish Vaswani and others from the Google Brain team.