**Prompt Engineering Guide**

**Principles, Techniques, and Applications to Harness the Power of Prompts in LLMs as a Data Analyst**

[Large Language Model (LLM)](https://www.youtube.com/watch?v=iR2O2GPbB0E) is on the rise, driven by the popularity of [ChatGPT by OpenAI](https://openai.com/blog/chatgpt) which took the internet by storm. As a practitioner in the data field, I look for ways to best utilize this technology in my work, especially for insightful-yet-practical work as a Data Analyst.

LLMs can solve tasks without additional model training via “prompting” techniques, in which the **problem is presented to the model as a text prompt**. Getting to “**the right prompts**” are important to ensure the model is providing high-quality and accurate results for the tasks assigned.

In this article, I will be sharing the principles of prompting, techniques to build prompts, and the roles Data Analysts can play in this “prompting era”.

**What is prompt engineering?**

Quoting [Ben Lorica from Gradient Flow](https://gradientflow.com/the-future-of-prompt-engineering-getting-the-most-out-of-llms/), “**prompt engineering is the art of crafting effective input prompts to elicit the desired output from foundation models**.” It’s the iterative process of developing prompts that can effectively leverage the capabilities of existing generative AI models to accomplish specific objectives.

Prompt engineering skills can help us understand the capabilities and limitations of a large language model. **The prompt itself acts as an input to the model, which signifies the impact on the model output.** A good prompt will get the model to produce desirable output, whereas working iteratively from a bad prompt will help us understand the limitations of the model and how to work with it.

Isa Fulford and Andrew Ng in the [ChatGPT Prompt Engineering for Developers course](https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/) mentioned two main principles of prompting:

* Principle 1: **Write clear and specific instructions**
* Principle 2: **Give the model time to “think”**

I think prompting is *like giving instructions to a naive “machine kid”*.

The child is very intelligent, but you need to be **clear about what you need** from it (by providing explanations, examples, specified output format, etc) and **give it some space to digest and process it** (specify the problem-solving steps, ask it to slowly process it). The child, given its exposure, can also be *very creative and imaginary* in providing answers — which we call a [hallucination of the LLM](https://thestack.technology/the-big-hallucination-large-language-models-consciousness/). Understanding the context and providing the right prompt might help in avoiding this problem.

**Prompt Engineering Techniques**

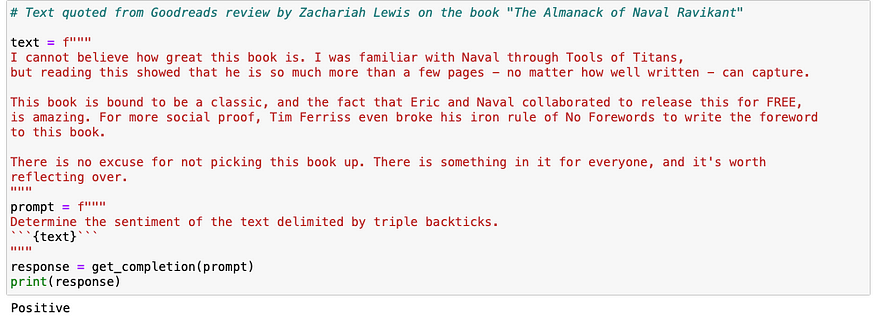
Prompt engineering is a growing field, with research on this topic rapidly increasing from 2022 onwards. Some of the state-of-the-art prompting techniques commonly used include n-shot prompting, chain-of-thought (CoT) prompting, and generated knowledge prompting.

*A sample Python notebook demonstrating these techniques* *is shared under* [*this GitHub project.*](https://github.com/oliviatan29/prompt-engineering)

**1. N-shot prompting (Zero-shot prompting, Few-shot prompting)**

Known for its variation like Zero-shot prompting and Few-shot prompting, the N in N-shot prompting represents the number of “training” or clues given to the model to make predictions.

**Zero-shot prompting** is where a model makes predictions **without any additional training.** This works for common straightforward problems like classification (i.e. sentiment analysis, spam classification), text transformation (i.e. translation, summarizing, expanding), and simple text generation on which the LLM has been largely trained.



Zero-shot prompting: Straightforwardly ask the model on sentiment (Image by Author)

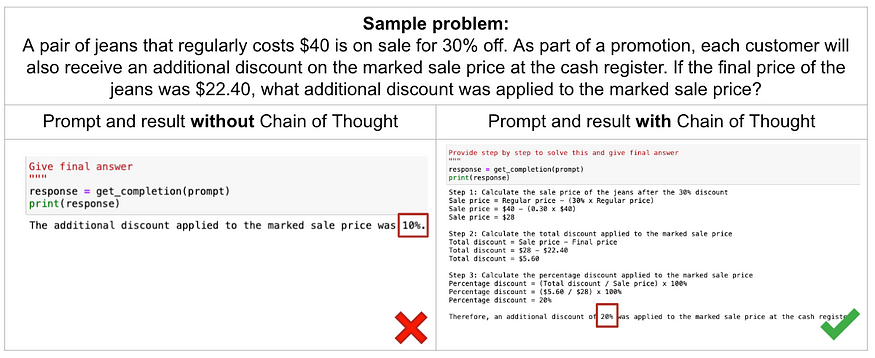
**Few-shot prompting** uses a **small amount of data** (typically between two and five) **to adapt its output** based on these small examples. These examples are meant to steer the model to better performance for a more context-specific problem.



Few-shot prompting: Give examples of how we expect the model output to be

**2. Chain-of-Thought (CoT) prompting**

[Chain-of-Thought prompting](https://ai.googleblog.com/2022/05/language-models-perform-reasoning-via.html) was introduced by Google researchers in 2022. In the *Chain-of-Thought prompting*, the model is prompted to **produce intermediate reasoning steps before giving the final answer** to a multi-step problem. The idea is that a model-generated chain of thought would mimic an intuitive thought process when working through a multi-step reasoning problem.



Chain-of-Thought prompting helps in driving the model to break down problems accordingly

This method enables models to decompose multi-step problems into intermediate steps, enabling them to solve complex reasoning problems that are not solvable with standard prompting methods.

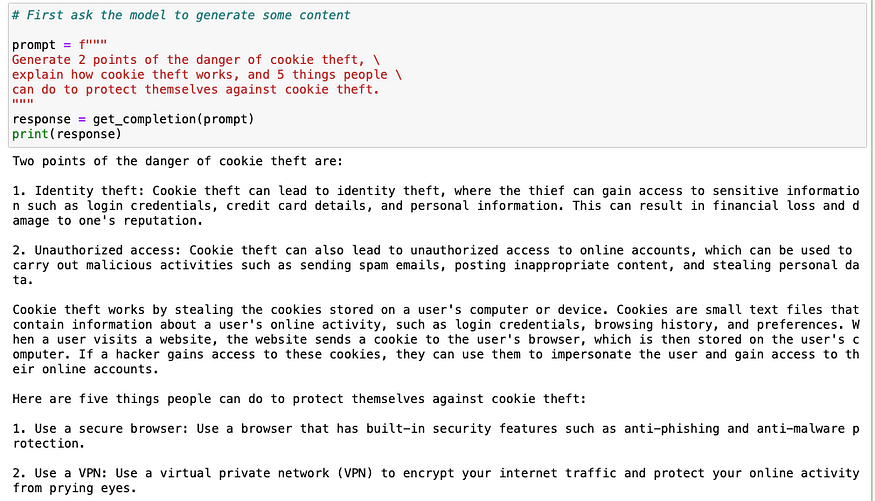
Some further variations of Chain-of Thought prompting include:

* [Self-consistency prompting](https://arxiv.org/pdf/2203.11171.pdf): Sample multiple diverse reasoning paths and **select the most consistent answers**. By utilizing a majority voting system, the model can arrive at more accurate and reliable answers.
* [Least-to-Most prompting (LtM)](https://arxiv.org/abs/2205.10625): Specify the chain of thought to first break a problem into a series of simpler subproblems and then **solve them in sequence**. Solving each subproblem is facilitated by the answers to previously solved subproblems. This technique is inspired by real-world educational strategies for children.
* [Active Prompting](https://arxiv.org/abs/2302.12246): **Scaling the CoT approach** by determining which questions are the most important and helpful ones for human annotation. It first calculates the uncertainty among the LLM’s predictions, then select the most uncertain questions, and these questions are selected for human annotation before being put into a CoT prompt.

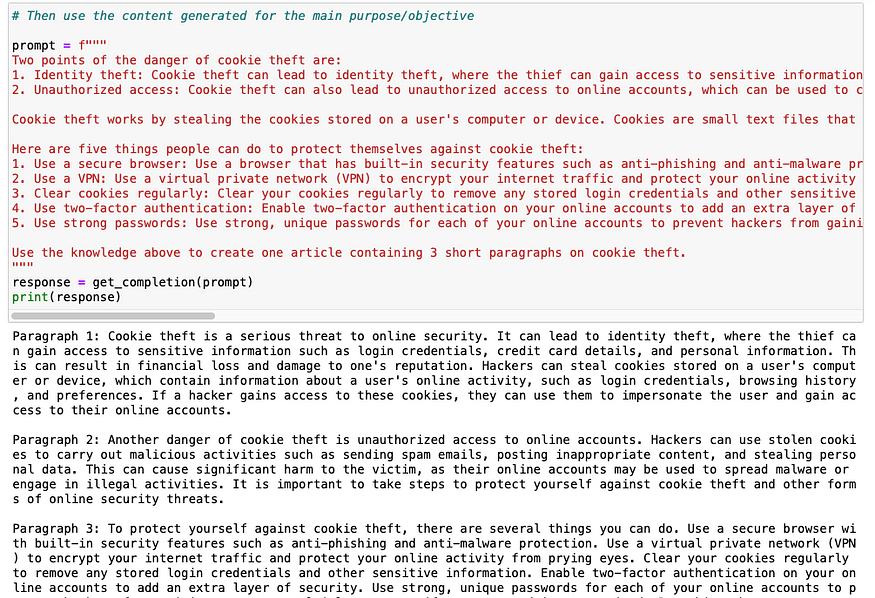
**3. Generated knowledge prompting**

The idea behind the [generated knowledge prompting](https://arxiv.org/pdf/2110.08387.pdf) is to ask the LLM to **generate potentially useful information** about a given question/prompt, and then **leverage that provided knowledge** as additional input for generating a final response.

For example, say you want to write an article about cybersecurity, particularly cookie theft. Before asking the LLM to write the article, you can ask it to generate some danger and protection against cookie theft. This will help the LLM write a more informative blog post.



Generated knowledge prompting: (1) Ask the model to generate some content



Generated knowledge prompting: (2) Use the generated content as input to the model

**Additional tactics**

On top of the above-specified techniques, you can also use these tactics below to make the prompting more effective

* **Use delimiters** like triple backticks (```), angle brackets (<>), or tags (<tag> </tag>) to indicate distinct parts of the input, making it cleaner for debugging and avoiding prompt injection.
* **Ask for structured output** (i.e. HTML/JSON format), this is useful for using the model output for another machine processing.
* Specify the **intended tone of the text** to get the tonality, format, and length of model output that you need. For example, you can instruct the model to formalize the language, generate not more than 50 words, etc.
* Modify the model’s **temperature parameter** to play around the model’s degree of randomness. The higher the temperature, the model’s output would be random than accurate, and even hallucinate.

*A sample Python notebook demonstrating these techniques* *is shared under* [*this GitHub project.*](https://github.com/oliviatan29/prompt-engineering)

**The Role of Analysts in Prompt Engineering**

As you can possibly infer from the examples above, prompt engineering requires a very specific technical communication craft. While you still require business context and problem-solving skills, it is still a new kind of craft that is not entirely covered as part of a [conventional data analytics skillset](https://medium.com/towards-data-science/how-does-a-remarkable-data-analyst-look-like-dd0e4326c670).

Data Analysts can leverage their context knowledge, problem-solving skills, and statistical/technical capabilities, with the addition of effective communication for prompt engineering. These are the key tasks related to prompt engineering (and LLMs) which potentially be done by Analysts:

* **Specifying LLM problems to be solved**. With an understanding of the LLM concepts, we can define the actions to be executed by the model (i.e. whether it is text classification, generation, or transformation problem) and the right question with reference points to be put as the prompts.
* **Iterative prompting**. In developing a data model, oftentimes we go through an iterative process. After building the initial model, we evaluate the result, refine it, and retry it along the way. Similarly for a prompt, we analyze where the result does not give what you want, and refine it with clearer instructions, additional examples, or specified steps. This requires critical reasoning which most Data Analysts are already good at.
* **Prompt versioning and management**. With iterative prompting, you will end up with numerous prompt attempts, and the identified model capabilities and/or limitations. It is important to keep track of and document these findings for team learning and continuous improvement, as with any other existing data analysis.
* **Designing for safe-prompting**. Although it has shown impressive capabilities, LLM is still in a very early stage and is prone to loopholes and limitations. There is this [hallucination problem](https://thestack.technology/the-big-hallucination-large-language-models-consciousness/) where models provide highly misleading information, and also [prompt injection](https://simonwillison.net/2023/Apr/14/worst-that-can-happen/) risk of getting untrusted text is used as part of the prompt. Depending on the use case of the model and prompting, Analysts can advise programmatic safeguards to limit the prompt usage and analysis of problematic prompting detection.

On top of leveraging the existing skills, Analysts need to **hone their communication skills and the ability to break down problems** to provide better prompts.

**Conclusion**

Large Language Models have shown promising results in performing numerous types of language tasks, and **prompt engineering is the key to unlocking these capabilities**. Prompt engineering is about communicating effectively with an AI to achieve desired results.

Several techniques can be used to do prompt engineering, but the foundational principle is consistent. It is about providing clear instructions to the model and helping it in digesting and processing these instructions. **Data Analysts can leverage their context knowledge and problem-solving skills to frame the right prompts and leverage their technical capabilities for designing prompt safeguards**.

For further resources on prompt engineering, check out:

* [ChatGPT Prompt Engineering for Developers](https://learn.deeplearning.ai/chatgpt-prompt-eng) (DeepLearning.AI course)
* [LearnPrompting.org](https://learnprompting.org/docs/intro)
* [PromptingGuide.ai](https://www.promptingguide.ai/)
* [OpenAI applications documentation](https://platform.openai.com/examples)
* [The Future of Prompt Engineering: Getting The Most Out of LLMs](https://gradientflow.com/the-future-of-prompt-engineering-getting-the-most-out-of-llms/) (Gradient Flow article)

I believe this area will grow even further in the next few years, and I’m excited to see and take part in the evolution.

**Guía de Ingeniería de Prompts**

**Principios, Técnicas y Aplicaciones para Aprovechar el Poder de los Prompts en LLMs como** Analista de Datos

Los Modelos de Lenguaje de Gran Tamaño (LLMs, por sus siglas en inglés) están en aumento, impulsados por la popularidad de ChatGPT de OpenAI, que ha causado sensación en Internet. Como profesional en el campo de los datos, busco formas de utilizar de la mejor manera esta tecnología en mi trabajo, especialmente para tareas perspicaces y prácticas como Analista de Datos. Los LLMs pueden resolver tareas sin necesidad de entrenamiento adicional mediante técnicas de "prompts", en las cuales se presenta el problema al modelo como un prompt de texto. Encontrar los "prompts adecuados" es importante para asegurar que el modelo proporcione resultados de alta calidad y precisión para las tareas asignadas.

En este artículo, compartiré los principios de los prompts, técnicas para construir prompts y los roles que los Analistas de Datos pueden desempeñar en esta "era de los prompts".

**¿Qué es la ingeniería de prompts?**

Citando a Ben Lorica de Gradient Flow, "la ingeniería de prompts es el arte de crear prompts de entrada efectivos para obtener la salida deseada de los modelos base". Es el proceso iterativo de desarrollar prompts que puedan aprovechar eficazmente las capacidades de los modelos generativos de IA existentes para lograr objetivos específicos.

Las habilidades de ingeniería de prompts pueden ayudarnos a comprender las capacidades y limitaciones de un modelo de lenguaje de gran tamaño. El prompt en sí actúa como una entrada al modelo, lo cual tiene un impacto en la salida del modelo. Un buen prompt hará que el modelo produzca una salida deseable, mientras que trabajar de forma iterativa desde un prompt deficiente nos ayudará a comprender las limitaciones del modelo y cómo trabajar con él.

Isa Fulford y Andrew Ng, en el curso "ChatGPT Prompt Engineering for Developers", mencionaron dos principios principales de los prompts:

* Principio 1: Escribir instrucciones claras y específicas.
* Principio 2: Dar tiempo al modelo para "pensar".

Creo que hacer prompts es como dar instrucciones a un "niño máquina" ingenuo. El niño es muy inteligente, pero necesitas ser claro sobre lo que necesitas de él (proporcionando explicaciones, ejemplos, formato de salida especificado, etc.) y darle espacio para digerirlo y procesarlo (especificar los pasos para resolver el problema, pedirle que lo procese lentamente). El niño, dada su exposición, también puede ser muy creativo e imaginativo al proporcionar respuestas, lo que llamamos alucinación del LLM. Comprender el contexto y proporcionar el prompt correcto puede ayudar a evitar este problema.

**Prompt Engineering Techniques**

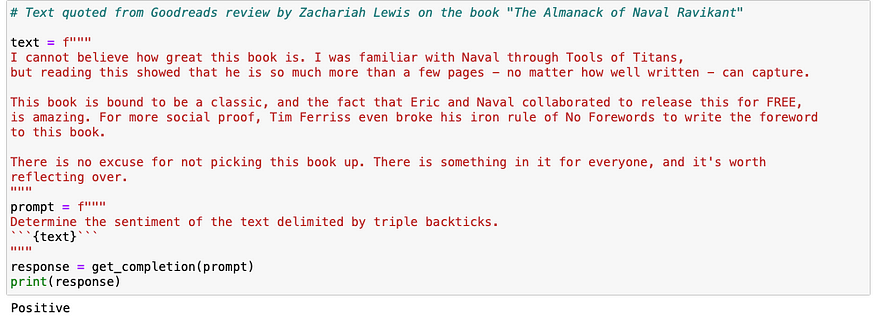
La ingeniería de instrucciones es un campo en crecimiento, con investigaciones sobre este tema aumentando rápidamente a partir de 2022. Algunas de las técnicas de instrucciones de última generación comúnmente utilizadas incluyen la instrucción n-shot, la instrucción de cadena de pensamiento (CoT) y la instrucción de conocimiento generada.

Se comparte un cuaderno de Python de muestra que demuestra estas técnicas en este proyecto de GitHub.

1. **Instrucción n-shot (Instrucción de cero disparos, Instrucción de pocos disparos)**

Conocida por sus variaciones como instrucción de cero disparos e instrucción de pocos disparos, la "N" en la instrucción n-shot representa el número de "entrenamientos" o pistas proporcionadas al modelo para realizar predicciones.

La instrucción de cero disparos es cuando un modelo realiza predicciones sin ningún entrenamiento adicional. Esto funciona para problemas comunes y directos como clasificación (por ejemplo, análisis de sentimiento, clasificación de spam), transformación de texto (por ejemplo, traducción, resumen, ampliación) y generación de texto simple en los cuales el modelo de lenguaje ha sido ampliamente entrenado.



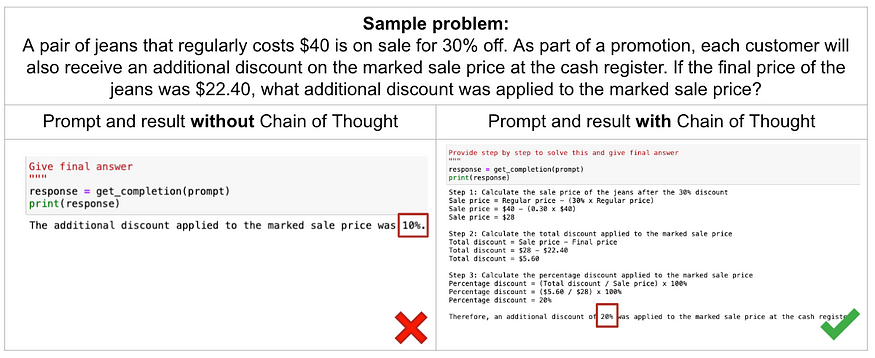
Instrucción de cero disparos: Pregunta directamente al modelo sobre el sentimiento (Imagen por el autor) La instrucción de pocos disparos utiliza una pequeña cantidad de datos (normalmente entre dos y cinco) para adaptar su resultado basándose en estos pequeños ejemplos. Estos ejemplos están destinados a guiar al modelo hacia un mejor rendimiento para un problema más específico en cuanto al contexto.



Instrucción de pocos disparos: Proporciona ejemplos de cómo esperamos que sea el resultado del modelo.

1. **Prompting de Cadena de Pensamiento (CoT)**

El prompting de Cadena de Pensamiento fue introducido por investigadores de Google en 2022. En el prompting de Cadena de Pensamiento, se le pide al modelo que produzca pasos intermedios de razonamiento antes de dar la respuesta final a un problema de múltiples pasos. La idea es que una cadena de pensamiento generada por el modelo imitaría un proceso de pensamiento intuitivo al resolver un problema de razonamiento de múltiples pasos.



El prompting de Cadena de Pensamiento ayuda a guiar al modelo a descomponer los problemas de manera adecuada.

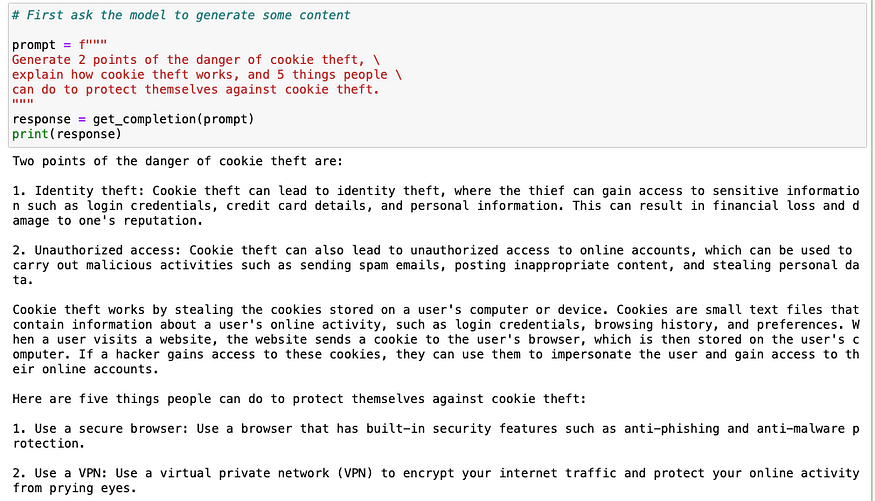
Este método permite a los modelos descomponer problemas de múltiples pasos en pasos intermedios, lo que les permite resolver problemas de razonamiento complejos que no son resolubles con métodos de prompting estándar. Algunas variaciones adicionales del prompting de Cadena de Pensamiento incluyen:

* Prompting de autoconsistencia: Muestrea múltiples caminos de razonamiento diversos y selecciona las respuestas más consistentes. Utilizando un sistema de votación mayoritaria, el modelo puede llegar a respuestas más precisas y confiables.
* Prompting de Menos a Más (LtM): Especifica la cadena de pensamiento para descomponer un problema en una serie de subproblemas más simples y luego resolverlos en secuencia. La resolución de cada subproblema se facilita mediante las respuestas a los subproblemas resueltos previamente. Esta técnica está inspirada en estrategias educativas del mundo real para niños.
* Prompting Activo: Escala el enfoque de CoT al determinar qué preguntas son las más importantes y útiles para la anotación humana. Primero calcula la incertidumbre entre las predicciones del LLM, luego selecciona las preguntas más inciertas y estas preguntas se seleccionan para la anotación humana antes de incluirlas en un prompt de CoT.

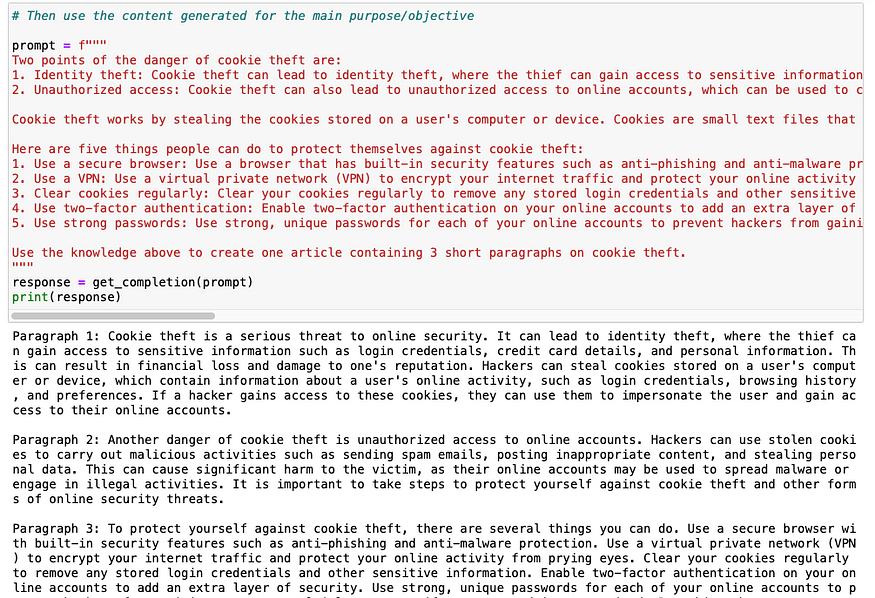
1. **Prompting de conocimiento generado**

La idea detrás del prompting de conocimiento generado es pedirle al LLM que genere información potencialmente útil sobre una pregunta o prompt determinado, y luego aprovechar ese conocimiento proporcionado como entrada adicional para generar una respuesta final.

Por ejemplo, digamos que quieres escribir un artículo sobre ciberseguridad, específicamente sobre el robo de cookies. Antes de pedirle al LLM que escriba el artículo, puedes pedirle que genere información sobre los peligros y la protección contra el robo de cookies. Esto ayudará al LLM a escribir una publicación de blog más informativa.



Prompting de conocimiento generado: (1) Pídele al modelo que genere algún contenido.



Prompting de conocimiento generado: (2) Utiliza el contenido generado como entrada para el modelo.

**Tácticas adicionales:**

Además de las técnicas mencionadas anteriormente, también puedes utilizar las siguientes tácticas para hacer que el prompting sea más efectivo:

* Utiliza delimitadores como triples acentos graves (```), corchetes angulares (<>) o etiquetas (<tag> </tag>) para indicar partes distintas de la entrada, lo que facilitará la depuración y evitará la inyección de prompts.
* Solicita una salida estructurada en formato HTML/JSON. Esto es útil para utilizar la salida del modelo en otro proceso de máquina.
* Especifica el tono deseado del texto para obtener la tonalidad, formato y longitud de la salida del modelo que necesitas. Por ejemplo, puedes instruir al modelo para formalizar el lenguaje, generar no más de 50 palabras, etc.
* Modifica el parámetro de temperatura del modelo para ajustar el grado de aleatoriedad. Cuanto mayor sea la temperatura, más aleatoria será la salida del modelo, e incluso puede llegar a ser alucinante.

Se comparte un ejemplo de cuaderno de Python que demuestra estas técnicas en este proyecto de GitHub.

**El papel de los analistas en el prompting de ingeniería**

Como puedes inferir de los ejemplos anteriores, el prompting de ingeniería requiere un oficio de comunicación técnica muy específico. Aunque aún se requiere contexto empresarial y habilidades para resolver problemas, es un tipo de oficio nuevo que no está completamente cubierto como parte de un conjunto de habilidades analíticas de datos convencional.

Los analistas de datos pueden aprovechar sus conocimientos de contexto, habilidades para resolver problemas y capacidades estadísticas/técnicas, junto con una comunicación efectiva para el prompting de ingeniería. Estas son las tareas clave relacionadas con el prompting de ingeniería (y LLMs) que potencialmente pueden ser realizadas por los analistas:

* Especificar los problemas de LLM a resolver. Con una comprensión de los conceptos de LLM, podemos definir las acciones a ejecutar por el modelo (por ejemplo, si se trata de un problema de clasificación, generación o transformación de texto) y la pregunta correcta con puntos de referencia que se utilizarán como prompts.
* Prompting iterativo. En el desarrollo de un modelo de datos, a menudo pasamos por un proceso iterativo. Después de construir el modelo inicial, evaluamos el resultado, lo refinamos y lo volvemos a probar en el camino. De manera similar, con un prompt, analizamos dónde el resultado no ofrece lo que deseamos y lo refinamos con instrucciones más claras, ejemplos adicionales o pasos específicos. Esto requiere razonamiento crítico en el que la mayoría de los analistas de datos ya son buenos.
* Versionado y gestión de prompts. Con el prompting iterativo, terminarás con numerosos intentos de prompts y las capacidades y/o limitaciones del modelo identificadas. Es importante realizar un seguimiento y documentar estos hallazgos para el aprendizaje en equipo y la mejora continua, al igual que con cualquier otro análisis de datos existente.
* Diseño para un prompting seguro. Aunque ha demostrado capacidades impresionantes, LLM aún se encuentra en una etapa muy temprana y es propenso a lagunas y limitaciones. Existe el problema de alucinación, donde los modelos proporcionan información muy engañosa, y también el riesgo de inyección de prompts al utilizar texto no confiable como parte del prompt. Dependiendo del caso de uso del modelo y del prompting, los analistas pueden aconsejar salvaguardias programáticas para limitar el uso de prompts y la detección de problemas problemáticos en el prompting.

Además de aprovechar las habilidades existentes, los analistas deben perfeccionar sus habilidades de comunicación y la capacidad de descomponer problemas para proporcionar mejores prompts.

**Conclusión**

Los Modelos de Lenguaje de Gran Escala han demostrado resultados prometedores en la realización de numerosos tipos de tareas de lenguaje, y la ingeniería de prompts es la clave para desbloquear estas capacidades. La ingeniería de prompts se trata de comunicarse de manera efectiva con una IA para lograr los resultados deseados.

Se pueden utilizar varias técnicas para realizar la ingeniería de prompts, pero el principio fundamental es consistente. Se trata de proporcionar instrucciones claras al modelo y ayudarlo a digerir y procesar estas instrucciones. Los analistas de datos pueden aprovechar sus conocimientos de contexto y habilidades para resolver problemas para crear los prompts adecuados y utilizar sus capacidades técnicas para diseñar salvaguardias de prompts.

Para obtener más recursos sobre ingeniería de prompts, consulta:

* Curso "ChatGPT Prompt Engineering for Developers" de DeepLearning.AI
* LearnPrompting.org
* PromptingGuide.ai
* Documentación de aplicaciones de OpenAI
* Artículo "The Future of Prompt Engineering: Getting The Most Out of LLMs" de Gradient Flow

Creo que esta área seguirá creciendo en los próximos años y estoy emocionado de ver y participar en su evolución.