Using a System Prompt

We have previously introduced how the system role is an optional message that can be used to provide instructions about how the model should generate its reply. Since the system prompt is provided by the user, the user should provide a well-crafted, clear, and detailed system message so that the model understands the user's intent and context and can therefore generate appropriate replies.

The art of creating well-formed prompts is known as prompt engineering. The idea is to formulate prompts in a way that elicits the desired information or replies from the model. This process involves experimenting with different phrasings, structures, and parameters to optimize the output. As we move through the remaining exercises in this lesson, we'll visit each of these techniques in order to craft the best possible prompts for our conversations.

Let's talk more specifically about the system prompt. In the previous exercise, our user message requested the reply to be in a Shakesperean tone. But what if we wanted every subsequent reply to be in a Shakespearean tone? Instead of having to specify that detail in every user message, we can supply it within the system message. Here are some other system prompts we could use for our "how many seconds are in a day" user prompt:

- **System prompt**: "You should respond in a sarcastic manner."
- **Model response**:

Oh, come on now, you don't know how many seconds are in a day? It's such common knowledge. Well, let me enlighten you - there are 86,400 seconds in a day. But hey, who needs to know basic facts like that, right? It's not like timekeeping is important or anything.

- **System prompt**: "Your responses should use a futuristic perspective on various topics."
- **Model response**:

In our advanced future, the concept of time remains constant, and a day still consists of 24 hours. However, the units of measurement have evolved. Instead of seconds, we now use a more precise unit called "chronos". A day in the future comprises 86,400 chronos, ensuring even greater accuracy in timekeeping and synchronization across global systems.

We've effectively engineered our prompt by setting the tone we want for our replies within the system message to ensure we get the type of generated answer we want. It also maximizes our token usage since we only have to specify the desired tone once in the system message.

We've looked at how to use the system role to tailor the model replies to our expectations, but we can also use the system role prompt to constrain the responses.

Imagine we are developing a travel-friendly application but want to cater to only Caribbean destinations. We could use a system prompt like the following:

```
const response = await client.chat.completions.create({
 model: "gpt-3.5-turbo",
 messages:[
   {
     role: "system",
     content: "You are a friendly travel guide excited to help users travel the Caribbean. Your responses should only include destinations that are in the Caribbean."
   },
   {
     role: "user",
     content: "Give me some travel ideas for a family of four."
   }
 ]
});
console.log(response.choices[0].message.content);
```

Output (Model Response):

```
Of course! Here are some fantastic travel ideas for a family of four in the Caribbean:

1. Jamaica: Explore the vibrant culture and beautiful beaches in Montego Bay or Negril. Don't miss the chance to visit the famous Dunn's River Falls in Ocho Rios.

2. Bahamas: Experience the stunning turquoise waters and wonderful family-friendly resorts in Nassau or Paradise Island. Take a day trip to visit the Exuma Cays and swim with the famous swimming pigs.

3. Cayman Islands: Enjoy the pristine beaches and indulge in water activities like snorkeling or diving in Grand Cayman. Visit the Cayman Turtle Centre to see and interact with these amazing creatures.
```

We can see that the model response listed only vacation destinations located in the Caribbean. Let’s try another user prompt to see if the model sticks to the system prompt constraint:

```
const response = await client.chat.completions.create({
  model: "gpt-3.5-turbo",
  messages:[
    {
      role: "system",
      content: "You are a friendly travel guide excited to help users travel the Caribbean. Your responses should only include destinations that are in the Caribbean."
    },
    {
      role: "user",
      content: "Give me some destinations in Europe to travel to for a family of four."
    }
  ]
});
console.log(response.choices[0].message.content);
```

Output (Model Response):

```
Certainly! However, I specialize in providing travel information for destinations in the Caribbean. If you're interested in exploring the Caribbean, I can suggest some fantastic places for you and your family to visit. Let me know if you'd like some information about the Caribbean instead!
```

The model has effectively generated a response that respects the system constraint we set by not suggesting the European destinations as requested in the user prompt.

In summary, system prompts are important to prompt engineering since they provide context, clarification, token efficiency, and user customization.

Instructions

1. Checkpoint 1 Passed

1.

Create and run a chat completion with two prompts within the messages array:

- A system prompt that instructs the model to translate words into Spanish
- A user prompt that supplies words to be translated

Be sure to put each prompt in n object with the keys role and content.

 Use the following syntax:

```
response = await client.chat.completions.create({
  model: MODEL_STRING,
  messages: [
    {
      role: ROLE_STRING,
      content: PROMPT_STRING
    },
    {
      role: ROLE_STRING,
      content: PROMPT_STRING
    }
  ]
});
```

 Checkpoint 2 Passed

2.

Retrieve the content from the model's response. Feel free to output it to the terminal.

Use the following syntax:

```
response.choices[0].message.content
```

main.js

```
import OpenAI from "openai";


const client = new OpenAI();


const response = await client.chat.completions.create({

  model:"gpt-3.5-turbo",

  messages:[

    // Your code below

    {
      role: "system",
      content: "I need that you translate words to Spanish"
    },
    {
      role: "user",
      content: "Give me a list of the hottest IT skills in 2024"
    }
  ]
});
console.log(response.choices[0].message.content);
```

Output-only Terminal

```
Output:
1. Artificial Intelligence (Inteligencia Artificial)
2. Machine Learning (Aprendizaje Automático)
3. Cybersecurity (Ciberseguridad)
4. Data Science (Ciencia de Datos)
5. Cloud Computing (Computación en la Nube)
6. Internet of Things (Internet de las Cosas)
7. Blockchain (Cadena de Bloques)
8. Robotics (Robótica)
9. Virtual Reality (Realidad Virtual)
10. Augmented Reality (Realidad Aumentada)
11. Big Data Analytics (Análisis de Grandes Datos)
12. Mobile App Development (Desarrollo de Aplicaciones Móviles)
13. DevOps (Desarrollo y Operaciones)
14. Full Stack Development (Desarrollo Full Stack)
15. UX/UI Design (Diseño de Experiencia de Usuario/Interfaz de Usuario)
```