Conceptos clave

Lo primero que me quedó claro fue qué es un **prompt**: básicamente, una instrucción que le doy a un modelo de lenguaje para que genere una respuesta. Esta instrucción define el contexto, el tono y la forma de la salida. Descubrí que los prompts no son simplemente texto: están hechos de **tokens**, y cada modelo tiene un límite para la cantidad que puede procesar. Por ejemplo, GPT-3.5 tiene una ventana de contexto de 4 096 tokens.

También aprendí a manejar parámetros importantes:

- **Temperatura**: controla qué tan aleatoria es la respuesta. Una temperatura baja da respuestas más predecibles; una alta, más creativas.
- **max_tokens**: limita el tamaño de la respuesta.

Estos elementos son clave para ajustar la calidad, creatividad y precisión de las salidas.

Diseño de prompts efectivos

Fui probando distintas formas de estructurar los prompts y llegué a una fórmula que me funciona bien:

1. **Mensaje del sistema**: uso esta parte para definir el rol del modelo. Ejemplo: "Eres un asistente técnico que responde con claridad y precisión."
2. **Instrucción clara**: le digo exactamente qué quiero. Ejemplo: "Resume este texto en 150 palabras con lenguaje neutro."
3. **Ejemplos (few-shot)**: si el modelo necesita entender el estilo que quiero, incluyo pares de ejemplo entrada–salida.

Este enfoque me ayudó a obtener respuestas más coherentes, especialmente cuando era necesario mantener un estilo específico o seguir instrucciones detalladas.

Técnicas de razonamiento avanzadas

Exploré técnicas que permiten guiar mejor al modelo cuando las tareas son más complejas:

- **Zero-Shot Prompting**: el modelo responde sin necesidad de ejemplos.
- **Few-Shot Prompting**: útil cuando se quiere que el modelo aprenda de ejemplos específicos.
- **Chain-of-Thought**: le pido al modelo que explique paso a paso antes de dar la respuesta. Esto mejora bastante la lógica de lo que genera.
- **Self-Consistency**: genero varias respuestas y escojo la que sea más coherente entre ellas.

Gran parte de estos experimentos los probé con ejercicios y ejemplos disponibles en PromptingGuide.ai y los recursos abiertos de DAIR.AI.

Seguridad y prevención de errores

Uno de los riesgos que encontré es el **prompt injection**: un ataque en el que el usuario maliciosamente altera la lógica del prompt. Leí estudios sobre esto en Arxiv y aprendí prácticas para mitigarlo:

- Sanitizar entradas.
- Limitar instrucciones del usuario.
- Validar outputs.

IBM también tiene buenas recomendaciones sobre esto, como parametrizar variables y aplicar controles de acceso.

Automatización y flujos reutilizables

Me metí con **LangChain** y usé su herramienta **LangSmith** para versionar y depurar mis prompts. Ahí pude montar workflows completos que incluyen pasos antes y después del prompt, lo que ayuda a automatizar procesos y mantener todo organizado.

También experimenté con cómo almacenar y recuperar prompts según el contexto, lo cual me pareció clave para proyectos a largo plazo.

Cómo mido si un prompt está funcionando

Aprendí a usar algunas métricas para medir si el output generado se parece a lo que espero:

- **BLEU** y **ROUGE**: miden similitud entre textos.
- **BERTScore**: compara los significados usando embeddings.

A esto le sumo una evaluación humana: comparo si la respuesta realmente cumple con la intención del prompt. Uso ambos métodos para ajustar y mejorar iterativamente.

Reforzando lo aprendido con Anki

Configuro tarjetas Anki para asegurarme de que la teoría no se me olvide:

- Tarjetas de definición: me ayudan a memorizar conceptos como "¿Qué hace la temperatura en un prompt?"
- Tarjetas con huecos (cloze): ideales para recordar fragmentos de estructuras complejas.

Uso el algoritmo **FSRS** para espaciar los repasos y retener mejor la información. Esta técnica me ha servido para que los términos técnicos y estrategias se me queden grabadas.

Proyecto práctico

Para aplicar lo aprendido, hice un chatbot en Replit con un enfoque corporativo:

1. Configuré un mensaje del sistema formal.

2. Combiné técnicas como few-shot y chain-of-thought.
3. Generé varias versiones de respuesta y elegí la mejor con self-consistency.
4. Medí los resultados con métricas automáticas para ver si podía mejorarlo.

Fue un experimento útil que me permitió unir la teoría con la práctica.

Comunidad y recursos que me ayudaron

Participé activamente en el servidor de Discord de **DAIR.AI**, donde encontré ejemplos, debates y consejos actualizados. También seguí blogs como Moon Technolabs y Starmorph que explican trucos útiles y ofrecen guías aplicables.

Además, consulté el centro de ayuda de OpenAI para entender el comportamiento de sus modelos y actualizaciones recientes.