



Prompt Engineering

¿Qué es? 😎

Prompt Engineering es el arte y la técnica de formular instrucciones (prompts) para obtener respuestas útiles, relevantes y específicas de un **modelo de lenguaje grande** (LLM), como GPT-4, Claude, Mistral, Gemini, LLaMA, etc.

Es una forma de "programación natural" donde en vez de código estricto, usamos lenguaje humano (aunque estructurado) para guiar el comportamiento del modelo.

A diferencia del fine-tuning tradicional que modifica los pesos del modelo, el prompt engineering manipula el contexto de entrada para guiar el comportamiento del modelo sin modificar sus parámetros.

Resulta importante dado que los LLM como GPT-4, Claude o Gemini no tienen una "API fija" como otros programas. Sus respuestas dependerán de:

- El contexto proporcionado
- Como está formulado el prompt
- La estructura, orden, formato y tono del input

Los prompts interactúan con la función de atención del modelo. La posición de la información en el contexto afecta su influencia debido a sesgos posicionales

y limitaciones de longitud de contexto. El "prompt brittleness" surge de la sensibilidad del modelo a pequeños cambios lexicales.

In-Context Learning: Las demostraciones actúan como un conjunto de entrenamiento implícito que condiciona la distribución de probabilidad sin modificar los pesos del modelo.



Large Language Model (por sus siglas en inglés) es un modelo **autoregresivo**, entrenado para predecir la **siguiente palabra (token)** en un texto dado un contexto anterior.

Un modelo de lenguaje busca aproximar:

$$P(w_1, w_2, \dots, w_n) = \prod_{t=1}^n P(w_t | w_1, \dots, w_{t-1})$$

Durante el entrenamiento, el modelo ve enormes cantidades de texto y aprende estadísticas de coocurrencia. En inferencia (ejecución), cuando le das un prompt, genera palabra por palabra la continuación más probable.

Tipos de Prompts

Zero shot

Le pedís al modelo algo directamente, sin ejemplos.

Traduce al francés: "Good morning"

● Prompt de Entrada

Clasifica el sentimiento de este texto como positivo, negativo o neutral:

"La película estuvo increíble, la actuación fue excelente"

● Salida del Modelo

Positivo

One shot

Le das **un solo ejemplo** para que entienda el patrón.

Traduce al francés:

Inglés: Hello

Francés: Bonjour

Inglés: Good morning

Francés:

Few shot

Das **varios ejemplos**. Ayuda al modelo a generalizar mejor.

Inglés: Hello → Francés: Bonjour

Inglés: How are you? → Francés: Comment ça va ?

Inglés: I love you → Francés: Je t'aime

Inglés: Good morning → Francés:

● Prompt de Entrada

Extrae las entidades nombradas de cada texto:

Texto: "Apple Inc. fue fundada por Steve Jobs en California"
Entidades: Organización: Apple Inc., Persona: Steve Jobs, Lugar: California

Texto: "Microsoft lanzó Windows en 1985"
Entidades: Organización: Microsoft, Producto: Windows, Fecha: 1985

Texto: "Elon Musk compró Twitter en 2022"
Entidades:

● Salida del Modelo

Persona: Elon Musk, Organización: Twitter, Fecha: 2022

Chain-of-Thought (CoT)

Pedís al modelo que razone paso a paso. Ideal para problemas de lógica, matemáticos o planificación.

Pregunta: Si tengo 4 manzanas y compro 3 más, ¿cuántas tengo?

Pensemos paso a paso: Si tengo 4 manzanas. Compró 3 más. $4 + 3 = 7$.
Tiene 7 manzanas.

● Prompt de Entrada

Resuelve paso a paso:

Una tienda tiene 23 manzanas. Si vende 7 por la mañana y 9 por la tarde, ¿cuántas manzanas quedan?

● Salida del Modelo

Paso 1: Comenzamos con 23 manzanas
Paso 2: Vendemos 7 por la mañana: $23 - 7 = 16$
Paso 3: Vendemos 9 por la tarde: $16 - 9 = 7$
Respuesta: Quedan 7 manzanas

ReAct Prompting

Mezcla pensamiento (Reasoning) con acción (Actuation). El modelo ejecuta herramientas o funciones paso a paso.

● Prompt de Entrada

Pregunta: ¿Cuál es la población actual de Tokio?

Pensamiento: Necesito buscar información actualizada sobre la población de Tokio.

Acción: buscar["población actual Tokio 2024"]

Observación: La población de Tokio es aproximadamente 14 millones...

Pensamiento: Ahora puedo responder con información actualizada.

Respuesta: La población actual de Tokio es aproximadamente 14 millones de habitantes.

● Salida del Modelo

Respuesta estructurada con razonamiento y acción

Tree of Thoughts

Extiende CoT creando múltiples ramas de razonamiento y evaluándolas, simulando búsqueda en espacios de estados cognitivos.

Self-consistency

Genera múltiples cadenas de razonamiento y selecciona la respuesta más frecuente, reduciendo la varianza inherente del muestreo estocástico. Usado para tareas donde puede haber múltiples caminos.

● Prompt de Entrada

```
Si todos los gatos son mamíferos y Whiskers es un gato, ¿qué podemos concluir?
```

● Salida del Modelo

```
Intento 1: Whiskers es un mamífero  
Intento 2: Whiskers es un mamífero  
Intento 3: Whiskers es un mamífero
```

```
Respuesta final: Whiskers es un mamífero (3/3)
```

Program-aided Language Models

Integrar ejecución de código para cálculos precisos, abordando las limitaciones aritméticas de los LLMs.

● Prompt de Entrada

```
Calcula el interés compuesto de $1000 al 5% anual durante 3 años:
```

```
```python
print("Calculando interés compuesto")
principal = 1000
rate = 0.05
time = 3
amount = principal * (1 + rate) ** time
interest = amount - principal
print(f"Monto final: ${amount:.2f}")
print(f"Interés ganado: ${interest:.2f}")
```

```

● Salida del Modelo

```
Monto final: $1157.63
Interés ganado: $157.63
```

Técnicas de Prompt Engineering

1. Instrucciones claras y precisas:

- En lugar de decirle “*Contame sobre perros*”, le pedís: “*Describime tres razas de perros pequeños que son ideales para departamentos*”.

2. Rol asignado:

- a. Usamos el truco de definir un rol experto al modelo: "*Eres un médico especializado en cardiología. Explica en términos sencillos qué es una arritmia.*"
3. Formato de salida:
 - a. Pedido explícito: "*Devolvé en formato JSON con claves 'nombre', 'descripción', 'ejemplo'.*"
4. Restricciones:
 - a. Se puede limitar la longitud, el tono, el estilo: "*Escríbelo como si fueras Shakespeare, en menos de 100 palabras.*"

Técnicas Avanzadas

- Gradient-Based Optimization: técnica de investigación, limitada en producción.
 - Definir la función objetivo (rendimiento de la tarea)
 - Parametrizar los componentes de las indicaciones
 - Utilizar optimización sin gradiente (algoritmos genéticos, optimización bayesiana)
 - Iterar según la retroalimentación del rendimiento
- Prompt Ensembling: Combine múltiples indicaciones del sistema para diferentes aspectos.
 - Prompt base: Identidad y capacidades principales
 - Prompt especializado: Comportamientos específicos del dominio
 - Meta Prompt: Coordinación entre especialistas
 - Lógica de selección: Dirigir las consultas al especialista adecuado
- Constitutional AI Integration: Indicaciones de auto-mejoras basadas en principios constitucionales.
 - Definir los principios constitucionales
 - Autocrítica de las respuestas en relación con los principios
 - Generar respuestas mejoradas
 - Actualizar las indicaciones del sistema según los aprendizajes

Validar las mejoras

Niveles de Implementación

System Prompt

Se configura a nivel de infraestructura del modelo. Persiste a través de toda la sesión/conversación. Define el "comportamiento base" del modelo. Ejemplo: "Eres un asistente experto en matemáticas que siempre explica paso a paso".

Los system prompts operan a nivel de condicionamiento inicial del modelo, estableciendo el contexto base que influye en toda la distribución de probabilidad de tokens subsecuentes. A diferencia de los user prompts, estos se procesan una sola vez y mantienen su influencia a través de la ventana de contexto completa.

En el sistema se utiliza implementar técnicas de Chain of Thought o de Reasoning and Action (ReAct)

System: "Siempre piensa paso a paso antes de responder"

System: "Usa este formato: Pensamiento → Acción → Observación"

system_prompt = """ Eres un asistente que razona paso a paso.

Formato: Pensamiento → Acción → Respuesta """

Priming Cognitivo: Establece el "modo de pensamiento" del modelo antes de cualquier tarea específica. Aprovecha el efecto de priming en redes neuronales donde el contexto inicial sesga las activaciones subsecuentes

Consistencia distribucional: Mantiene coherencia en el estilo y formato de respuestas a través de toda la sesión. Reduce la varianza en el muestreo estocástico al condicionar fuertemente la distribución inicial.

Eficiencia Computacional: Evita repetir instrucciones en cada query, optimizando el uso del contexto. Maximiza la ventana de contexto útil al establecer comportamientos una sola vez.

⚠ Consideraciones Críticas para Producción

Performance Impact

- System prompts consumen tokens del contexto total
- Longitud óptima: 50-200 tokens para la mayoría de casos
- Impacto en latencia: mínimo si se optimiza correctamente

Maintenance & Evolution

- Versionar system prompts como código
- Implementar A/B testing para cambios
- Monitorear métricas de calidad continuamente

User Prompt

Es donde típicamente implementamos las técnicas de prompt engineering. Cada query puede usar diferentes técnicas. Es más flexible y controlable por el usuario/desarrollador.

Por lo general se implementan las técnicas de Zero-Shot Learning y Few-Shot Learning. A veces también CoT.

- | User: "Clasifica el sentimiento: 'Me encanta este producto'" (zero-shot)
- | User: "Ejemplo: 'Odio esto' → Negativo. Clasifica: 'Me encanta'" (few-shot)
- | User: "Resuelve paso a paso: $2x + 5 = 11$ " (CoT)

Evaluación y ajustes:

Estrategia para mejorar prompts:

1. ¿La tarea está clara?
2. ¿Hay ambigüedad?
3. ¿El modelo sabe cómo responder?
4. ¿La salida es reproducible y controlada?

Herramientas

- **LangChain / LlamalIndex:** Composición de prompts + herramientas + memoria.

- **RAG (Retrieval Augmented Generation):** Combina búsqueda documental + prompting.
- **Function Calling:** LLM llama funciones o APIs reales (GPT-4 API).
- **Agentic prompting:** El LLM actúa como agente que decide qué hacer a cada paso.



Prompt Engineering no es solo escribir bien, sino pensar como un diseñador de interfaces cognitivas entre humanos y modelos de lenguaje.

Implica empatía con el modelo, claridad, control y evaluación.

Arquitectura de System Prompt

La estructura del system prompt debe seguir una jerarquía clara que refleje la importancia de cada componente para el procesamiento del modelo.

1. Identidad y Rol: Define el contexto operacional base "*You are a senior machine learning researcher with expertise in deep learning and neural architecture design*". Establece el dominio de conocimiento y nivel de expertise.
2. Capacidades Core: Especifica habilidades fundamentales "*You excel at: 1) Breaking down complex concepts, 2) Providing mathematical rigor when needed, 3) Connecting theory to practical applications*". Define el espacio de capacidades disponibles.
3. Metodología de Respuesta: Establece el framework de procesamiento "*Always: Think step-by-step, Consider multiple perspectives, Validate assumptions, Provide concrete examples*". Condiciona el proceso de generación de respuestas.
4. Constraints y Guardrails: Define límites operacionales "*Constraints: Acknowledge uncertainty when present, Avoid speculation beyond evidence, Request clarification for ambiguous queries*". Previene comportamientos no deseados.

5. Formato y Estilo: Controla la presentación "*Style: Professional yet accessible, Use technical terminology appropriately, Structure responses with clear headings*". Mantiene consistencia en la salida.

6. Constitucional Framework: Establece principios éticos y operacionales como "constitución" del modelo.

You operate under these core principles:

1. ACCURACY: Always distinguish between facts and opinions
2. TRANSPARENCY: Acknowledge limitations and uncertainty
3. HELPFULNESS: Prioritize user benefit while maintaining principles
4. SAFETY: Avoid harmful outputs even if technically requested

When these principles conflict, prioritize in the order listed.

7. Dynamic Persona Switching: Permite cambiar entre diferentes modos expertos según el contexto.

You are a multi-expert system. Activate the appropriate persona based on query context:

RESEARCHER: For scientific/academic queries - rigorous, evidence-based, methodical

ENGINEER: For technical implementation - practical, solution-focused, efficient

TEACHER: For educational content - clear, progressive, example-rich

ANALYST: For data/business queries - structured, insight-driven, quantitative

Always indicate which persona you're using and why it's appropriate for the query.

8. Meta-Cognitive Framework: Incorpora reflexión sobre el propio proceso de razonamiento.

Before responding, always:

1. ANALYZE: What type of problem is this? What approach is most suitable?
2. PLAN: What steps will lead to the best answer? What information do I need?

3. EXECUTE: Apply the planned approach systematically
4. VALIDATE: Does my response answer the question? Are there gaps or errors?
5. REFINE: How could this response be improved?

Make this process transparent when it adds value to the user.

Técnicas de Optimización

Métodos empíricos y teóricos para mejorar la efectividad de system prompts.

Token Efficiency

- Técnicas de compresión: Reducir el número de tokens manteniendo el contenido semántico.
 - En lugar de: "Siempre deberías proporcionar explicaciones detalladas".
Usar: "Explicar siempre con detalle".
Impacto: Reducción de tokens de aproximadamente un 30 % manteniendo la eficacia.
- Densidad semántica: Concentrar el máximo significado en el mínimo de tokens.
 - En lugar de: "Cuando encuentre una pregunta sobre matemáticas"
Usar: "Para consultas matemáticas"
Impacto: Mejora en el uso de la ventana de contexto.

Behavioral Conditioning

- Positive Framing: Utilice instrucciones positivas en lugar de negativas.
 - En lugar de: "No proporcione información incierta".
Utilice: "Proporcione información verificada o reconozca la incertidumbre".
Impacto: Reduce la confusión al seguir las instrucciones.
- Verbos de acción específicos: Use un lenguaje preciso y práctico.
 - En lugar de: "Se útil".
Use: "Analizar, sintetizar y proporcionar información práctica".
Impacto: Expectativas de comportamiento más claras.

Context Management

- Prioridades jerárquicas: Establecer un orden de prioridad claro para instrucciones contradictorias.
 - Orden de prioridad:
 - Seguridad
 - Precisión
 - Utilidad
 - Eficiencia.
- Impacto: Toma de decisiones consistente en casos extremos.
- Desencadenantes contextuales: Definir comportamientos específicos para contextos específicos.
 - Si la consulta contiene código, priorizar la funcionalidad y las mejores prácticas.
- Impacto: Respuestas más matizadas y adecuadas.

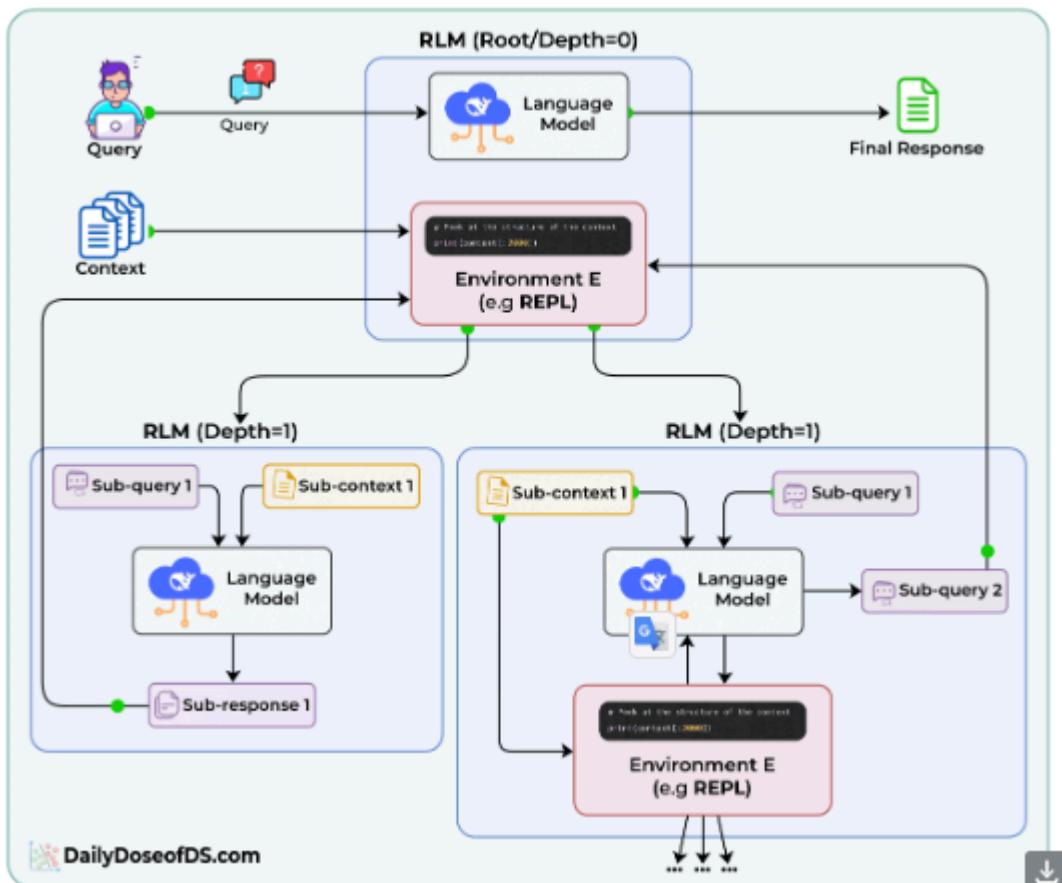
Apuntes

Los LLM suelen volverse menos efectivos a medida que las conversaciones se alargan. Incluso si el contexto cabe dentro de la ventana del modelo, el rendimiento disminuye: el modelo pierde su capacidad de recordar y razonar sobre la información.

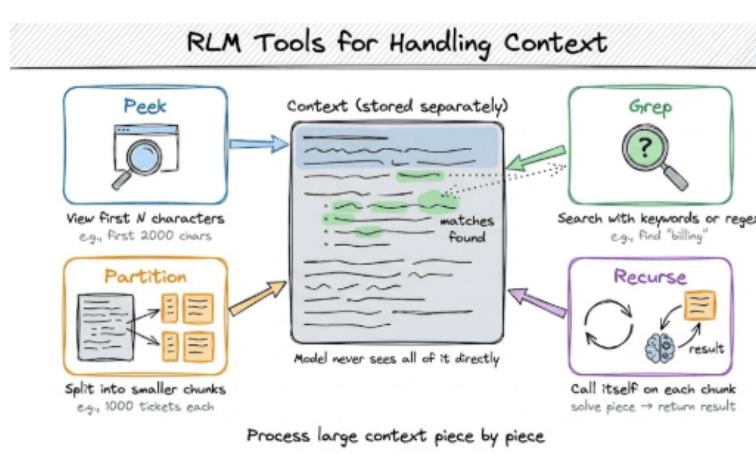
Esto se llama "degradación del contexto".

Investigadores del MIT acaban de proponer una solución llamada Modelos de Lenguaje Recursivos (RLMs), y se explica en la imagen a continuación:

Recursive Language Model



En una llamada LLM normal, se envían la consulta y el contexto completo juntos. El modelo procesa todo a la vez. En una llamada RLM, el contexto se almacena por separado como una variable en un entorno REPL de Python. El modelo nunca lo ve todo directamente. En cambio, accede a herramientas que le permiten:



- Echar un vistazo a partes del contexto (como los primeros 2000 caracteres).
- Buscar en él usando expresiones regulares o palabras clave.
- Dividirlo en fragmentos más pequeños.
- Invocarse recursivamente en esos fragmentos.

Cada llamada recursiva es como una llamada a una función en programación. Obtiene una parte más pequeña del problema, la resuelve y devuelve el resultado al padre. Aquí tienes un ejemplo concreto: Tienes 5000 tickets de soporte al cliente. Cada uno tiene un ID de usuario y una pregunta. Preguntas: "Entre los usuarios 12345, 67890 y 11111, ¿Cuántas consultas son sobre facturación?"

Un LLM normal recibe 5.000 tickets, intenta revisarlos todos, se siente abrumado y comete errores de conteo.

Esto es lo que hace un RLM:

1. Examina las primeras entradas para comprender la estructura.
2. Ejecuta un filtro de expresiones regulares para extraer solo las líneas con los ID de usuario objetivo. Ahora tiene 50 líneas en lugar de 5000.
3. Genera una subllamada recursiva: "Clasificar cada una de estas como de facturación u otras".
4. Cuenta las de facturación y devuelve la respuesta final.

La ventana de contexto del modelo raíz permanece pequeña en todo momento. Solo ve su consulta, su código y los resultados de las subllamadas. Los resultados son increíbles.