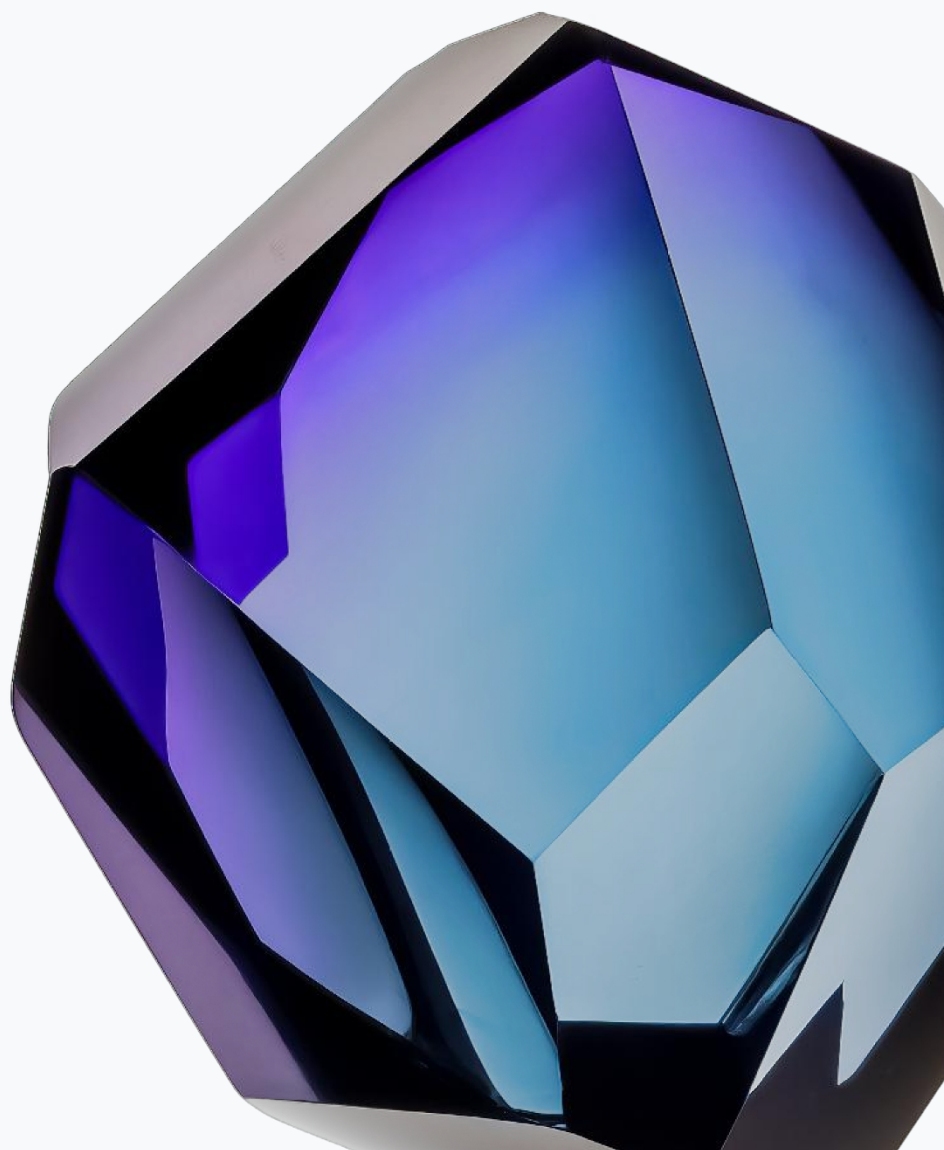


# Ingeniería Prompt

Autor: Lee Boonstra



## **Agradecimientos**

### **Colaboradores de contenido**

Michael Sherman

Yuan Cao

Erick Armbrust

Anant Nawalgaria

Antonio Gulli

Simone Cammel

### **Comisarios y editores**

Antonio Gulli Anant

Nawalgaria Grace

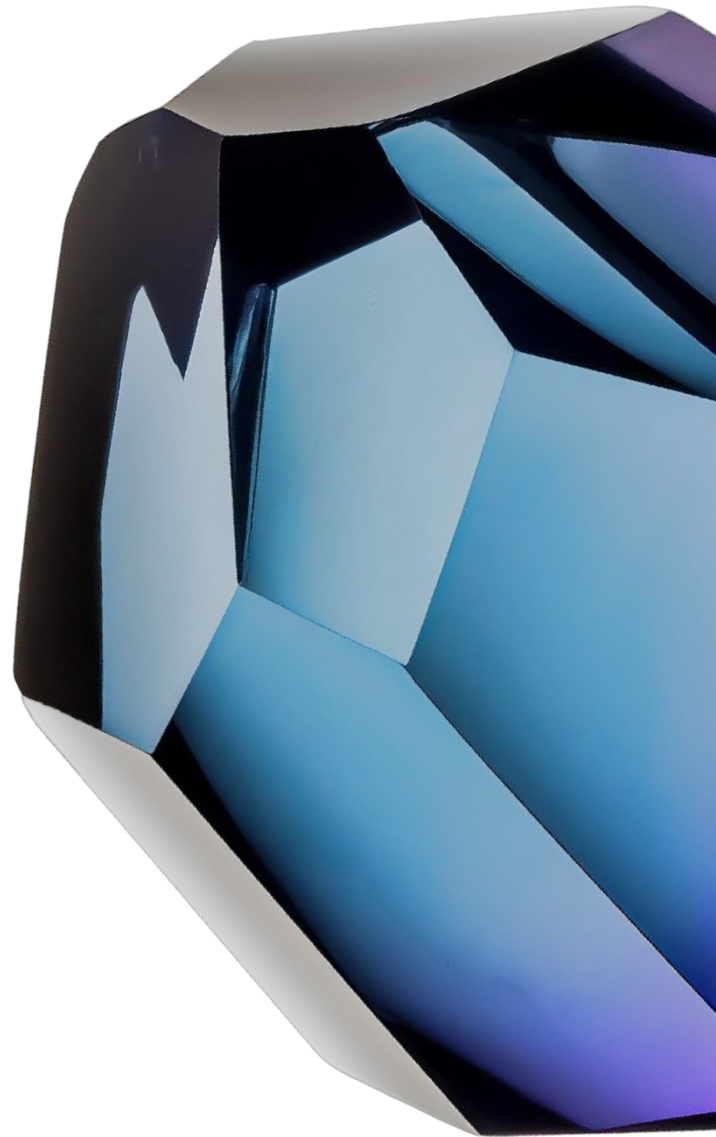
Mollison

### **Redactor técnico**

Joey Haymaker

### **Diseñador**

Michael Lanning




# Índice

Introducción .....	6
Ingeniería Prompt .....	7
Configuración de salida LLM .....	8
Longitud de salida .....	8
Controles de muestreo .....	9
Temperatura .....	9
Top-K y top-P .....	10
Puesta en común .....	11
Técnicas de incitación .....	13
Incitación general / tiro por cero .....	13
Un disparo y pocos disparos .....	15
Sistema, contexto y funciones .....	18
Indicaciones del sistema .....	19
Indicación de funciones .....	21
Indicaciones contextuales .....	23



Preguntar con la mano .....	25
Cadena de pensamiento (CdT) .....	29
Autoconsistencia .....	32
Árbol de los pensamientos (ToT) .....	36
ReAct (razonar y actuar) .....	37
Automatic Prompt Ingeniería .....	40
Indicación de código .....	42
Sugerencias para escribir código .....	42
Preguntas para explicar el código .....	44
Indicaciones para traducir código .....	46
Sugerencias para depurar y revisar el código .....	48
¿Qué pasa con el estímulo multimodal? .....	54
Buenas prácticas .....	54
Proporcione ejemplos .....	54
Diseño con sencillez .....	55
Especifique el resultado .....	56
Utilizar instrucciones en lugar de restricciones .....	56
Control de la longitud máxima de las fichas .....	58
Utilizar variables en las instrucciones .....	58
Experimentar con formatos de entrada y estilos de escritura .....	59
Para tareas de clasificación con pocos disparos, mezcle las clases .....	59
Adaptarse a las actualizaciones de los modelos .....	60
Experimente con los formatos de salida .....	60

Reparación de JSON	61
Trabajar con esquemas	62
Experimentar con otros ingenieros	63
CoT Buenas prácticas	64
Documentar los distintos intentos de aviso	64
Resumen	66
Notas finales	68



No hace falta ser un científico de datos o un ingeniero de aprendizaje automático: todo el mundo puede escribir un aviso.

## Introducción

Cuando pensamos en la entrada y la salida de un gran modelo lingüístico, la entrada que utiliza el modelo es un texto ( veces acompañado de otras modalidades, como imágenes). para predecir un resultado específico. No es necesario ser un científico de datos o un ingeniero de aprendizaje automático: todo el mundo puede escribir una predicción. Sin embargo, elaborar la predicción más eficaz puede ser complicado. Muchos aspectos de su aviso afectan su eficacia: el modelo que utiliza, los datos de entrenamiento del modelo, las configuraciones del modelo, su elección de palabras, el estilo y el tono, la estructura y el contexto son todos importantes. Por lo tanto, la ingeniería de las instrucciones es un proceso iterativo. Unas instrucciones inadecuadas pueden dar lugar a respuestas ambiguas e imprecisas, y pueden obstaculizar la capacidad del modelo para proporcionar resultados significativos.

Cuando chateas con el chatbot Gemini<sup>(1)</sup> básicamente escribes prompts, sin embargo este whitepaper se centra en escribir prompts para el modelo Gemini dentro de Vefiex AI o usando la API, porque al prompting el modelo directamente tendrás acceso a la configuración como la temperatura, etc.

Este libro blanco trata en detalle la ingeniería de avisos. Analizaremos las diversas técnicas de prompting para ayudarle a conseguir stafiéd y compartir consejos y mejores prácticas para convertirse en un prompting expefi. También hablaremos de algunos de los retos a los que puede enfrentarse al crear prompts.

## ingeniería

Recuerde cómo funciona un LLM: es un motor de predicción. El modelo toma un texto secuencial como entrada y luego predice cuál debería ser el siguiente token, basándose en los datos con los que fue entrenado. El LLM está operacionalizado para hacer esto una y otra vez, añadiendo el token predicho previamente al final del texto secuencial para predecir el siguiente token. El siguiente La predicción de tokens se basa en la relación entre lo que hay en los tokens anteriores y lo que el LLM ha visto durante su entrenamiento.

Cuando se escribe un prompt, se intenta configurar el LLM para que prediga la secuencia correcta de tokens. La ingeniería de instrucciones es el proceso de diseñar instrucciones de alta calidad que guíen a los LLM para producir resultados precisos. Este proceso implica hacer ajustes para encontrar la mejor instrucción, optimizar la longitud de la instrucción y evaluar el estilo de escritura y la estructura de la instrucción en relación con la secuencia de palabras. a la tarea. En el contexto del procesamiento del lenguaje natural y los LLM, una indicación es una entrada que se proporciona al modelo para generar una respuesta o predicción.

Estas indicaciones pueden utilizarse para realizar diversos tipos de tareas de comprensión y generación, como resumen de textos, extracción de información, formulación de preguntas y respuestas, clasificación de textos, traducción de lenguajes o códigos, generación de códigos y documentación o razonamiento de códigos.

No dudes en consultar las guías de Google<sup>2,3</sup> con ejemplos sencillos y eficaces.

En la ingeniería de avisos, se empieza por elegir un modelo. Es posible que los avisos deban optimizarse para su modelo específico, independientemente de si utiliza modelos de lenguaje Gemini en Vertex AI, GPT, Claude o un modelo de código abierto como Gemma o LLaMA.

Además de la indicación también tendrás que jugar con las distintas configuraciones de un LLM.

## Salida LLM configuración

Una vez elegido el modelo, hay que determinar su configuración. La mayoría de los LLM vienen con varias opciones de configuración que controlan la salida del LLM. Una ingeniería rápida eficaz requiere establecer estas configuraciones de forma óptima para su tarea.

### Salida longitud

Un parámetro de configuración importante es el número de tokens a generar en una respuesta. Generar más tokens requiere más cálculos por parte del LLM, lo que conlleva un mayor consumo de energía, tiempos de respuesta potencialmente más lentos y costes más elevados.

Reducir la longitud de la salida del LLM no hace que el LLM sea más sucinto estilística o textualmente en la salida que crea, sólo hace que el LLM deje de predecir más tokens una vez alcanzado el límite. Si sus necesidades requieren una longitud de salida shofi, posiblemente también tendrá que diseñar su prompt para acomodarse.

La restricción de la longitud de salida es especialmente importante para algunas técnicas de avisos de LLM, como ReAct, en las que el LLM seguirá emitiendo tokens inútiles después de la respuesta deseada.

Ten en cuenta que generar más tokens requiere más cálculo por parte del LLM, lo que conlleva un mayor consumo de energía y tiempos de respuesta potencialmente más lentos, lo que se traduce en mayores costes.

## Muestreo controles

Los LLM no predicen formalmente una sola palabra. Más bien, los LLM predicen probabilidades de lo que podría ser el siguiente token, y cada token del vocabulario del LLM recibe una probabilidad. Estas probabilidades de los tokens se muestrean para determinar cuál será el siguiente token producido. Temperatura, top-K y top-P son los ajustes de configuración más comunes que determinan cómo se procesan las probabilidades de los tokens predichos para elegir un único token de salida.

## Temperatura

La temperatura controla el grado de aleatoriedad en la selección de fichas. Las temperaturas más bajas son buenas para las solicitudes que esperan una respuesta más determinista, mientras que las temperaturas más altas pueden conducir a resultados más diversos o inesperados. Una temperatura de 0 (descodificación codiciosa) es

determinista: siempre se selecciona la ficha de mayor probabilidad (aunque tenga en cuenta que si dos fichas tienen la misma probabilidad predicha más alta, dependiendo de cómo se implemente el desempate puede que no siempre obtenga la misma salida con temperatura 0).

Las temperaturas cercanas al máximo tienden a crear más resultados aleatorios. Y a medida que la temperatura aumenta, todas las fichas tienen la misma probabilidad de ser la siguiente ficha pronosticada.

El control de temperatura Gemini puede entenderse de forma similar a la función softmax utilizada en el aprendizaje automático. Un ajuste de temperatura bajo refleja una temperatura softmax baja ( $T$ ), haciendo hincapié en una única temperatura preferida con alta certeza. Un ajuste de temperatura Gemini más alto es como una temperatura softmax alta, haciendo que un rango más amplio de temperaturas en torno a más aceptable el ajuste seleccionado. Esta mayor flexibilidad se adapta a situaciones en las que una temperatura rígida y precisa puede no ser esencial como, por ejemplo, cuando se experimenta con resultados creativos.

## Top-K y top- P

Top-K y top-P (también conocido como muestreo de núcleo)<sup>4</sup>son dos configuraciones de muestreo utilizadas en los LLM para restringir la predicción del siguiente token para que proceda de los tokens con las probabilidades de predicción más altas. Al igual que la temperatura, estos ajustes de muestreo controlan la aleatoriedad y la diversidad del texto generado.

- El muestreo Top-K selecciona las K fichas más probables de la distribución prevista del modelo. Cuanto más alto sea el top-K, más creativo y variado será el resultado del modelo; cuanto más bajo sea el top-K, más inquieto y factual será el resultado del modelo. Un top-K de 1 equivale a una decodificación codiciosa.

- El muestreo **Top-P** selecciona los tokens superiores cuya probabilidad acumulada no supera un valor cefiain (P). Los valores de P van de 0 (descodificación avariciosa) a 1 (todos los tokens del vocabulario del LLM).

La mejor manera de elegir entre top-K y top-P es experimentar con ambos métodos (o con los dos juntos) y ver cuál produce los resultados que busca.

## **Pufling todo junto**

La elección entre top-K, top-P, temperatura y número de fichas a generar depende de la aplicación específica y del resultado deseado, y todos los ajustes se influyen mutuamente. También es importante asegurarse de que se entiende cómo el modelo elegido combina los diferentes ajustes de muestreo.

Si la temperatura, top-K y top-P están disponibles (como en Vefiex Studio), los tokens que cumplan los criterios top-K y top-P son candidatos para el siguiente token predicho, y entonces se aplica a la muestra de los tokens que han superado los criterios top-K y top-P. Si sólo se dispone de top-K o top-P, el comportamiento es el mismo pero sólo se utiliza el ajuste top-K o P.

Si la temperatura no está disponible, se seleccionan aleatoriamente las fichas que cumplan los criterios top-K y/o top-P para obtener una única ficha predicha.

En ajustes extremos de un valor de configuración de muestreo, ese ajuste de muestreo anula otros ajustes de configuración o se vuelve irrelevante.

- Si se fija la temperatura en 0, top-K y top-P se vuelven irrelevantes: la ficha más probable se convierte en la siguiente ficha pronosticada. Si la temperatura es muy alta (por encima de 1, por lo general en torno a los 10), la temperatura deja de tener importancia y cualquier ficha que haga a través de los criterios top-K y/o top-P se muestrean aleatoriamente para elegir la siguiente ficha pronosticada.
- Si se fija top-K en 1, la temperatura y top-P pasan a ser irrelevantes. Sólo un token pasa el criterio top-K, y ese token es el siguiente token predicho. Si se fija top-K muy alto, como el tamaño del vocabulario del LLM, cualquier token con una probabilidad distinta de cero de ser el siguiente token cumplirá los criterios de top-K y no se seleccionará ninguno.
- Si establece top-P en 0 (o un valor muy pequeño), la mayoría de las implementaciones de muestreo LLM sólo considerarán el token más probable para cumplir con los criterios top-P, haciendo que la temperatura y top-K sean irrelevantes. Si estableces top-P a 1, cualquier ficha con una probabilidad distinta de cero de ser la siguiente ficha cumplirá los criterios top-P, y ninguna será seleccionada.

Como punto general de stafiing, una temperatura de 0,2, un top-P de 0,95 y un top-K de 30 darán resultados relativamente coherentes que pueden ser creativos pero no . Si quiere resultados especialmente creativos, pruebe a stafiingar con una temperatura de 0,9, un top-P de 0,99 y un top-K de 40. Y si quieres resultados menos creativos, prueba a stafiing con una temperatura de 0,1, top-P de 0,9 y top-K de 20.

Por último, si su tarea siempre tiene una única respuesta correcta (por ejemplo, responder a un problema de matemáticas), stafi con una temperatura de 0.

**NOTA:** Con más libertad (mayor temperatura, top-K, top-P y tokens de salida), el LLM podría generar un texto menos relevante.

**ADVERTENCIA:** ¿Ha visto alguna vez una respuesta que termine con una gran cantidad de palabras de relleno? Esto también se conoce como el "error del bucle de repetición", que es un problema común en los Modelos de Lenguaje de Gran Tamaño en los que el modelo se queda atascado en un ciclo, generando repetidamente la misma palabra (de relleno), frase o estructura de oración, a menudo exacerbado por una temperatura inadecuada y top-k/

ajustes top-p. Esto puede ocurrir tanto a bajas como a altas temperaturas, aunque por razones diferentes. A bajas temperaturas, el modelo se vuelve excesivamente determinista, ciñéndose rígidamente a la ruta de mayor probabilidad, lo que puede conducir a un bucle si esa ruta vuelve a visitar el texto generado previamente. A la inversa, a temperaturas elevadas, la salida del modelo se vuelve excesivamente aleatoria, aumentando la probabilidad de una palabra o frase elegida al azar, por casualidad, a un estado anterior, creando un bucle debido al gran número de opciones disponibles. En ambos, el proceso de muestreo del modelo se "atasca", dando lugar a una salida monótona y poco útil hasta que se llena la ventana de salida. Para resolver este problema, a menudo es necesario modificar cuidadosamente la temperatura y los valores top-k/top-p para encontrar el equilibrio óptimo entre determinismo y aleatoriedad.

## Prompting técnicas

Los LLM están diseñados para seguir instrucciones y se entrenan con grandes cantidades de datos para que puedan entender una pregunta y generar una respuesta. Sin embargo, los LLM no son perfectos: cuanto más claro sea el texto, mejor podrá predecir el siguiente texto probable. Además, las técnicas específicas que aprovechan cómo se entrenan los LLM y cómo funcionan los LLM le ayudarán a obtener los resultados relevantes de los LLM.

Ahora que ya sabemos qué es la ingeniería de avisos y lo que implica, vamos a ver algunos ejemplos de las técnicas de avisos más importantes.

### **Incitación general / cero disparo**

Un aviso *de disparo cero*<sup>5</sup> es el tipo de aviso más sencillo. Sólo proporciona una descripción de una tarea y algún texto para que el LLM se familiarice con él. Este texto puede ser cualquier cosa: una pregunta, el guión de una historia o instrucciones. El nombre "zero-shot" significa "sin ejemplos".

Vamos a utilizar Vefiex AI Studio (para Lenguaje) en Vefiex AI,<sup>6</sup> que proporciona un patio de recreo para probar prompts. En la Tabla 1, verás un ejemplo de pregunta de tiro cero para clasificar críticas de cine.

El formato de tabla que se utiliza a continuación es una buena forma de documentar las instrucciones. Sus prompts probablemente pasarán por muchas iteraciones antes de que terminen en una base de código, por lo que es importante hacer un seguimiento de su trabajo de ingeniería de prompts de una manera disciplinada y estructurada. Más información sobre este formato de tabla, la importancia del seguimiento del trabajo de ingeniería de avisos y el proceso de desarrollo de avisos se encuentra en la sección de Mejores Prácticas más adelante en este capítulo ("Documentar los diversos intentos de avisos").

La temperatura del modelo debe fijarse en un número bajo, ya que no se necesita creatividad, y utilizamos los valores top-K y top-P por defecto de gemini-pro, que efectivamente desactivan ambos ajustes (ver 'Configuración de salida LLM' más arriba). Preste atención a la salida generada. Las palabras *perturbador* y *obra maestra* deberían complicar un poco la predicción, ya que ambas palabras se utilizan en la misma frase.

Nombre	1_1_clasificación_de_películas		
Objetivo	Clasificar las críticas de películas como positivas, neutras o		
Modelo	negativas. gemini-pro		
Temperatura	0.1	Límite de fichas	5
Top-K Prompt	N/A	Top-P	1
	<p>Clasifica las críticas de películas como POSITIVAS, NEUTRALES o NEGATIVAS. Crítica: "Her" es un inquietante estudio que revela la dirección en la que se dirige la humanidad si se permite que la IA siga evolucionando, sin control. Ojalá hubiera más películas como esta obra maestra. Sentimiento:</p>		
Salida	POSITIVO		

Tabla 1. Ejemplo de aviso de disparo cero

Cuando el "tiro por la culata" no funciona, se pueden hacer demostraciones o poner ejemplos, lo que da lugar a "tiros por la culata" y "tiros por la culata". Aviso general / tiro por cero

## Un disparo y pocos- disparo

A la hora de crear instrucciones para los modelos de IA, es útil proporcionar ejemplos. Estos ejemplos pueden ayudar al modelo a entender lo que se le pide. Los ejemplos son especialmente útiles cuando se quiere dirigir el modelo hacia una estructura o patrón de salida cefiain.

Un **estímulo "one-shot"** proporciona un *único* ejemplo, de ahí el nombre de "one-shot". La idea es que el modelo tenga un ejemplo que pueda imitar para completar mejor la tarea.

Un **aviso de pocos disparos**<sup>7</sup> proporciona *múltiples* ejemplos al modelo. Este enfoque muestra al modelo un patrón que debe seguir. La idea es similar a la de una sola vez, pero con múltiples ejemplos del patrón deseado aumenta la probabilidad de que el modelo siga el patrón.

El número de ejemplos necesarios depende de varios factores, como la complejidad de la tarea, la calidad de los ejemplos y las capacidades del modelo de IA generativa que se utilice. Como regla general, se deben utilizar al menos entre tres y cinco ejemplos. Sin embargo, puede que tengas que utilizar más ejemplos para tareas más complejas, o puede que tengas que utilizar menos debido a la limitación de longitud de entrada de tu modelo.

En la Tabla 2 se muestra un ejemplo de solicitud de unos pocos disparos, vamos a utilizar los mismos ajustes de configuración del modelo gemini-pro que antes, aparte de aumentar el límite de tokens para dar cabida a la necesidad de una respuesta más larga.

Objetivo	Parsear pedidos de pizza a		
Modelo	JSON gemini-pro		
Temperatura	0.1	Límite de fichas	250
Top-K	N/A	Top-P	1
Pregunta	<p>Convierte el pedido de pizza de un cliente en un JSON válido:</p> <p>EJEMPLO:</p> <p>Quiero una pizza pequeña con queso, salsa de tomate y pepperoni.</p> <p>Respuesta JSON:</p> <pre> ... {   "tamaño": "pequeño",   "tipo": "normal",   "ingredientes": [["queso", "salsa de tomate", "peperoni"]] } ... </pre> <p style="text-align: right;"><b>Continúa en la página siguiente...</b></p>		

<b>Pregunte a</b>	<p>EJEMPLO:</p> <p>¿Puedo pedir una pizza grande con salsa de tomate, albahaca y mozzarella</p> <pre>{   "tamaño": "grande",   "tipo": "normal",   "ingredientes": ["salsa de tomate", "bazel", "mozzarella"]} </pre> <p>Ahora, me gustaría una pizza grande, con la primera mitad de queso y mozzarella. Y la otra salsa de tomate, jamón y piña.</p> <p>Respuesta JSON:</p>
<b>Salida</b>	<pre>``` {   "tamaño": "grande",   "tipo": "mitad-mitad",   "ingredientes": ["queso", "mozzarella"], ["salsa de tomate",   "jamón", "piña"] } ```</pre>

Tabla 2. Un ejemplo de incitación de pocos disparos

Cuando elija ejemplos para su pregunta, utilice ejemplos que sean relevantes para la tarea que desea realizar. Los ejemplos deben ser diversos, de alta calidad y estar bien escritos. Un pequeño error puede confundir al modelo y dar lugar a resultados no deseados.

Si lo que se pretende es generar resultados robustos frente a una variedad de entradas, es importante incluir casos extremos en los ejemplos. Los casos extremos son entradas inusuales o inesperadas, pero que el modelo debe ser capaz de manejar.

## Sistema, contexto y función

El sistema, el contexto y el rol son técnicas utilizadas para guiar la generación de texto por parte de los LLM, pero se centran en aspectos diferentes:

- **Las instrucciones del sistema** establecen el contexto general y el objetivo del modelo lingüístico. Define el "panorama general" de lo que debe hacer el modelo, como traducir una lengua, clasificar una reseña, etc.
- **Las indicaciones contextuales** proporcionan detalles específicos o información de fondo relevante para la conversación o la tarea en curso. Ayuda al modelo a entender los matices de lo que se le pregunta y a adaptar la respuesta en consecuencia.
- **El rol** asigna al modelo lingüístico un carácter o identidad específicos que debe adoptar. Esto ayuda al modelo a generar respuestas coherentes con el rol asignado y sus conocimientos y comportamientos asociados.

Puede haber un solapamiento considerable entre los avisos de sistema, contextuales y de rol. Por ejemplo, un aviso que asigna una función al sistema también puede tener un contexto.

Sin embargo, cada tipo de indicador tiene un objetivo principal ligeramente diferente:

- **Indicación del sistema:** Define las capacidades fundamentales del modelo y su propósito general.
- **Indicación contextual:** Proporciona información inmediata y específica de la tarea para guiar la respuesta. Es muy específica para la tarea o entrada actual, que es dinámica.
- **Indicación de rol:** Enmarca el estilo de salida y la voz del modelo. Añade una capa de especificidad y personalidad.

Distinguir entre avisos del sistema, contextuales y de función proporciona un marco para diseñar avisos con una intención clara, permitiendo combinaciones flexibles y facilitando el análisis de cómo influye cada tipo de aviso en los resultados del modelo lingüístico.

Analicemos estos tres tipos de indicaciones.

## Sistema

El cuadro 3 contiene una pregunta del sistema, en la que especifico información adicional sobre cómo devolver la salida. Aumenté la temperatura para obtener un mayor nivel de creatividad y especifiqué un límite de tokens más alto. Sin embargo, debido a mi clara instrucción sobre cómo devolver la salida, el modelo no devolvió texto adicional.

Objeto	Clasifica las críticas de películas en positivas, neutras o		
Modelo	negativas. gemini-pro		
Temperatura	1	Límite de fichas	5
Top-K	40	Top-P	0.8
Pregunta a	<p>Clasifique las críticas de películas como positivas, neutras o negativas. Devuelve la etiqueta solo en mayúsculas.</p> <p>Crítica: "Her" es un inquietante estudio que revela la dirección en la que se dirige la humanidad si se permite que la IA siga evolucionando, sin control. Es tan perturbadora que no pude verla.</p>		
Salida	Sentimiento: NEGATIVO		

Tabla 3. Ejemplo de aviso del sistema

Los avisos del sistema pueden ser útiles para generar resultados que cumplan requisitos específicos. El nombre 'prompt del sistema' significa en realidad 'proporcionar una tarea adicional al sistema'. Por ejemplo, podrías usar un prompt del sistema para generar un fragmento de código compatible con un lenguaje de programación específico, o podrías usar un prompt del sistema para devolver una estructura cefiain. Echa un vistazo a la Tabla 4, donde devuelvo la salida en formato JSON.

<b>Objetivo</b>	Clasifica las críticas de películas como positivas, neutras o negativas,		
<b>Modelo</b>	devuelve JSON. gemini-pro		
<b>Temperatura</b>	1	<b>Límite de fichas</b>	1024
<b>Top-K</b>	40	<b>Top-P</b>	0.8
<b>Pregunta</b>	<p>Clasifica las críticas de películas como positivas, neutras o negativas. Devuelve un JSON válido:</p> <p>Crítica: "Her" es un inquietante estudio que revela la dirección en la que se dirige la humanidad si se permite que la IA siga evolucionando, sin control. Es tan perturbadora que no pude verla.</p> <p>Esquema:</p> <pre>     ...     PELÍCULA     A:     {       "sentimiento": Cadena "POSITIVO"   "NEGATIVO"   "NEUTRO",       "nombre": Cadena     }     CRÍTICAS DE CINE:     {       "movie_reviews": [MOVIE]     }     ...         </pre>		
<b>Salida</b>	<pre>     ...     Respuesta JSON:     {       "movie_reviews": [         {           "sentimiento":             "NEGATIVO", "nombre":             "Ella"         }       ]     }     ...         </pre>		

Tabla 4. Ejemplo de consulta del sistema en formato JSON

Hay algunos beneficios en devolver objetos JSON desde un prompt que extrae datos. En una aplicación del mundo real no necesito crear manualmente este formato JSON, ya puedo devolver los datos en un orden suavizado (muy útil cuando se trabaja con objetos datetime), pero lo más impofiantly, pidiendo un formato JSON obliga al modelo para crear una estructura y limitar las alucinaciones.

Los avisos del sistema también pueden ser realmente útiles para la seguridad y la toxicidad. Para controlar la salida, simplemente añada una línea adicional a su aviso como: 'Debe ser respetuoso en su respuesta'.

## **Papel**

El "Role prompting" es una técnica de ingeniería de avisos que consiste en asignar un papel específico al modelo de IA gen. Esto puede ayudar al modelo a generar resultados más relevantes e informativos, ya que el modelo puede elaborar sus respuestas para el papel específico que se le ha asignado.

Por ejemplo, puedes asignar a un modelo de IA gen la función de editor de libros, profesor de kindergarten u orador motivacional. Una vez que se ha asignado un papel al modelo, puedes darle instrucciones específicas para ese papel. Por ejemplo, puedes pedir a un profesor que cree un plan de clase que luego pueda revisar. Echa un vistazo a la Tabla 5, donde el modelo adopta el papel de guía de viajes.

Modelo de	Actúa como guía de viajes y proporciona 3		
objetivos	sugerencias de viaje gemini-pro		
Temperatura	1	Límite de fichas	1024
Top-K Prompt	40	Top-P	0.8
	<p>Quiero que hagas de guía de viajes. Te escribiré sobre mi ubicación y tú me sugerirás 3 lugares que visitar cerca de mí. En algunos casos, también les indicaré el tipo de lugares que visitaré.</p> <p>Mi sugerencia: "Estoy en Ámsterdam y sólo quiero visitar museos".</p> <p>Sugerencias de viaje:</p>		
Salida	<ol style="list-style-type: none"> <li>1. Rijksmuseum: Museo de fama mundial, conocido por obras maestras holandesas como "La ronda de noche" de Rembrandt. No se pierda la increíble colección de pinturas y artefactos que muestran el arte y la historia holandeses.</li> <li>2. Museo Van Gogh: Sumérgase en las cautivadoras obras de Vincent van Gogh. El museo alberga la mayor colección de sus pinturas y dibujos, entre ellos "La noche estrellada" y "Girasoles".</li> <li>3. Museo Stedelijk de Ámsterdam: Descubra arte moderno y contemporáneo de todo el mundo. La colección del museo incluye obras de Picasso, Kandinsky y Rietveld, albergadas en un llamativo edificio moderno.</li> </ol>		

Tabla 5. Ejemplo de indicación de funciones

El ejemplo anterior muestra un ejemplo en el que se adopta el papel de un agente de viajes. Si cambia el papel por el de profesor de geografía, observará que recibe una respuesta diferente.

Definir una perspectiva de rol para un modelo de IA le proporciona un modelo del tono, el estilo y la experiencia específica que busca para mejorar la calidad, la relevancia y la eficacia de sus resultados.

Aquí tienes algunos estilos entre los que puedes elegir y que me parecen eficaces:

Confrontativo, Descriptivo, Directo, Formal, Humorístico, Influyente, Informal, Inspirador, Persuasivo

Modifiquemos nuestra indicación del Cuadro 6 para incluir un estilo humorístico e inspirador.

<p><b>Pregunte a</b></p>	<p>Quiero que hagas de guía de viajes. Te escribiré sobre mi ubicación y tú me sugerirás 3 lugares que visitar cerca de mí en clave de humor.</p> <p>Mi sugerencia: "Estoy en Manhattan".</p> <p>Sugerencias de viaje:</p>
<p><b>Salida</b></p>	<ol style="list-style-type: none"> <li>1. Contemple el Empire State of Mind: Ascienda a las vertiginosas alturas del Empire State Building y disfrute de la gloria del horizonte de Manhattan. Prepárese para sentirse como King Kong en lo alto de la Gran Manzana, pero sin el plátano gigante del tamaño de un simio.</li> <li>2. Afíliate en el MoMA: Da rienda suelta al aficionado que llevas dentro en el Museo de Aficionados Modernos (MoMA). Contemple obras maestras que le dejarán perplejo. mente y te hacen cuestionar si tus dibujos de figuras de palo tienen algún mérito afiístico.</li> <li>3. Compre hasta caer rendido en la Quinta Avenida: Disfrute de una terapia de compras en la emblemática Quinta Avenida. Prepárate para un shock de precios mientras miras escaparates (o compras de verdad) en boutiques de diseñadores que harán llorar a tu cartera. Pero bueno, estás en Manhattan, ¡así que aprovecha la fabulosidad!</li> </ol>

Tabla 6. Un ejemplo de incitación al rol con tono y estilo humorísticos

## Contextual

Al proporcionar indicaciones contextuales, puede ayudar a garantizar que sus interacciones con la IA sean lo más fluidas y eficientes posible. El modelo podrá comprender más rápidamente su solicitud y generar respuestas más precisas y pertinentes, como puede ver en el ejemplo de la Tabla 7.

Modelo de	Sugiero afiicles para un blog sobre juegos retro		
objetivos	gemini-pro		
Temperatura	1	Límite de fichas	1024
Top-K Prompt	40	Top-P	0.8
Salida	<p>Contexto: Estás escribiendo para un blog sobre videojuegos arcade retro de los 80.</p> <p>Sugiera 3 temas sobre los que escribir un artículo con unas líneas de descripción de lo que debería contener dicho artículo.</p>		
	<ol style="list-style-type: none"> <li>1. <b>**La evolución del diseño de los recreativos</b> Este artículo explora la evolución de los diseños de los recreativos, desde los primeros de madera y metal de los años setenta hasta los elegantes con luces de neón de los ochenta y posteriores.</li> <li>2. <b>**Explosión del pasado: juegos arcade icónicos de los 80</b> - Este afiicle ofrece un vistazo a la historia, el impacto y el legado de algunos de los juegos arcade más emblemáticos e influyentes de los años ochenta, destacando sus innovaciones, mecánicas populares y encanto perdurable.</li> <li>3. <b>**El auge y el renacimiento retro de Pixel Afi:**</b> - Este afiicle ahondará en la evolución del pixel afi como estilo visual definitorio de los primeros juegos arcade, rastreando sus raíces, explorando sus técnicas y debatiendo el resurgimiento del pixel afi en los juegos modernos y en el afi digital.</li> </ol>		

Tabla 7. Ejemplo de estímulo contextual

## Paso atrás

El "paso atrás"<sup>8</sup> es una técnica para mejorar el rendimiento que consiste en pedir al LLM que considere primero una pregunta general relacionada con la tarea específica que se está realizando y, a continuación, introducir la respuesta a esa pregunta general en una pregunta posterior para la tarea específica. Este "paso atrás" permite al LLM activar conocimientos previos y procesos de razonamiento relevantes antes de intentar resolver el problema específico.

Al considerar los principios más amplios y subyacentes, los estudiantes de LLM pueden generar respuestas más precisas y perspicaces. Las instrucciones paso a paso animan a los LLM a pensar de forma crítica y a aplicar sus conocimientos de formas nuevas y creativas. Cambia la indicación final de la tarea al utilizar más conocimientos en los parámetros del LLM de los que entrarían en juego cuando se le indica directamente.

Puede ayudar a mitigar los sesgos en las respuestas del LLM, al centrarse en principios generales en lugar de en detalles concretos, el step-back prompting.

Echemos un vistazo a estos ejemplos para comprender cómo el step-back prompting puede mejorar los resultados. Analicemos primero un aviso tradicional (Tabla 8) antes de compararlo con un aviso de retroceso (Tabla 9):

Objetivo	Escribe un argumento para un nivel de un de disparos en primera persona.		
Modelo	gemini-pro		
Temperatura	1	Límite de fichas	1024
Top-K	40	Top-P	0.8
Pregunte a	Escribe un argumento de un párrafo para un nuevo nivel de un videojuego de disparos en primera persona que sea desafiante y atractivo.		
Salida	El nivel comienza con el escuadrón del jugador emboscado en una densa zona urbana por una implacable facción enemiga. Los jugadores deben abrirse paso a través de un laberinto de callejones y edificios en ruinas, utilizando el sigilo y la precisión para eliminar a las fuerzas enemigas y evitar ser detectados. El jugador debe utilizar el entorno a su favor, tendiendo trampas, cubriéndose y realizando movimientos calculados para sobrevivir. Por el camino, descubre información que sugiere que el enemigo está planeando un ataque devastador, y el jugador debe correr contrarreloj para reunir pruebas y transmitir las a sus jefes antes de que el enemigo pueda llevar a cabo sus planes.		

Tabla 8. Un prompt tradicional antes de compararlo con un prompt step back

Si ajustas la temperatura a 1, puedes acabar con todo tipo de escritos creativos para un argumento, pero también es bastante aleatorio y genérico. Así que demos un paso atrás:

<p><b>Pregunte a</b></p>	<p>Basados en populares juegos de acción en primera persona, ¿qué son 5 escenarios ficticios clave que contribuyen a una trama de niveles desafiante y atractiva en un de disparos en primera persona?</p>
<p><b>Salida</b></p>	<ol style="list-style-type: none"> <li>1. <b>**Base militar abandonada**</b>: Un extenso complejo militar postapocalíptico plagado de soldados mutados y robots rebeldes, ideal para desafiantes combates con armas de fuego.</li> <li>2. <b>**Ciudad cyberpunk**</b>: Un entorno urbano futurista, iluminado con luces de neón, con altísimos rascacielos y densos callejones, con enemigos mejorados cibernéticamente y mecánicas de pirateo.</li> <li>3. <b>**Nave espacial alienígena**</b>: Una enorme nave alienígena varada en Eafih, con pasillos espeluznantes, secciones de gravedad cero y criaturas extraterrestres con las que enfrentarse.</li> <li>4. <b>**Ciudad infestada de zombis**</b>: Una ciudad desolada invadida por hordas de zombis agresivos, con intensos combates cuerpo a cuerpo y resolución de puzles para encontrar un paso seguro.</li> <li>5. <b>**Instalación de investigación submarina**</b>: Un laboratorio de aguas profundas inundado de agua, lleno de criaturas acuáticas mutadas y que requiere habilidades de sigilo y exploración submarina para sobrevivir.</li> </ol>

Tabla 9. Ejemplo de ayuda a la coherencia

Sí, esos temas parecen encajar bien en un videojuego en primera persona. Volvamos al prompt original, pero esta vez incluyamos la respuesta del paso atrás como contexto y veamos qué nos devuelve.

<p><b>Pregunte a</b></p>	<p>Contexto: 5 temas atractivos para un de disparos en primera persona:</p> <ol style="list-style-type: none"> <li>1. <b>**Base militar abandonada**</b>: Un extenso complejo militar postapocalíptico repleto de soldados mutados y robots rebeldes, ideal para desafiantes combates con armas de fuego.</li> <li>2. <b>**Ciudad cyberpunk**</b>: Un entorno urbano futurista, iluminado con luces de neón, con altísimos rascacielos y densos callejones, con enemigos mejorados cibernéticamente y mecánicas de pirateo.</li> <li>3. <b>**Nave espacial alienígena**</b>: Una enorme nave alienígena varada en la Tierra, con pasillos espeluznantes, secciones de gravedad cero y criaturas extraterrestres a las que enfrentarse.</li> <li>4. <b>**Ciudad infestada de zombis**</b>: Una ciudad desolada invadida por hordas de zombis agresivos, con intensos combates cuerpo a cuerpo y resolución de puzzles para encontrar un paso seguro.</li> <li>5. <b>**Instalación de investigación submarina**</b>: Un laboratorio de aguas profundas inundado de agua, lleno de criaturas acuáticas mutantes y que requiere habilidades de sigilo y exploración submarina para sobrevivir.</li> </ol> <p>Toma uno de los temas y escribe un argumento de un párrafo para un nuevo nivel de un videojuego de disparos en primera persona que sea desafiante y atractivo.</p>
<p><b>Salida</b></p>	<p>En el fondo de un turbio abismo se encuentra una ruinoso instalación de investigación submarina, testimonio de la ambición humana y sus desastrosas consecuencias. Envuelta en la oscuridad, palpitante por el zumbido de la maquinaria averiada y atormentada por los ecos de las almas ahogadas, la instalación es ahora el caldo de cultivo de una pesadilla nacida de las profundidades marinas. El jugador, un marine de élite equipado con avanzados equipos de buceo y armamento experimental, tiene la misión de aventurarse en este reino acuático de terror. Armado con resiliencia e ingenio, deberá navegar por pasillos traicioneros, resolver enigmáticos rompecabezas y enfrentarse a espantosas monstruosidades marinas que acechan en las profundidades. Desde lidiar con la aplastante presión de las profundidades marinas hasta superar a los astutos depredadores acuáticos, cada momento en este submundo insaciable pone a prueba los límites de la resistencia y el valor humanos.</p>

Tabla 10. Ejemplo de ayuda a la coherencia

Parece un videojuego interesante. Utilizando técnicas de retroalimentación puedes aumentar la precisión de tus indicaciones.

## Cadena de pensamiento (CoT)

El estímulo de la Cadena de Pensamiento (CoT) <sup>9</sup> es una técnica para mejorar las capacidades de razonamiento de los LLM mediante la generación de **pasos intermedios de razonamiento**. Esto ayuda al LLM a generar respuestas más precisas. Se puede combinar con el prompting de pocos disparos para obtener mejores resultados en tareas más complejas que requieren razonar antes de responder ya que es un reto con una cadena de pensamiento de cero disparos.

CoT tiene muchas ventajas. En primer lugar, tiene un bajo coste y, al mismo tiempo, es muy eficaz y funciona bien con LLM estándar (por lo que no es necesario ajustarlo). También se obtiene interpretabilidad con los mensajes CoT, ya que se puede aprender de las respuestas del LLM y ver los pasos de razonamiento que se siguieron. Si hay algún fallo, podrás identificarlo. Aparece la cadena de pensamiento para mejorar la robustez al pasar de una versión LLM a otra. Lo que significa que el rendimiento de su indicador debería variar menos entre diferentes LLMs que si su indicador no utiliza cadenas de razonamiento. Por supuesto, también hay desventajas, pero son algo intuitivas.

La respuesta LLM incluye la cadena de razonamiento del pensamiento, lo que significa más fichas de salida, lo que significa que las predicciones cuestan más dinero y llevan más tiempo.

Para explicar el siguiente ejemplo de la Tabla 11, primero vamos a intentar crear un prompt que no utilice CoT prompting para mostrar los defectos de un modelo lingüístico grande.

Pregunte a	Cuando tenía 3 , mi pareja tenía 3 veces mi edad. tengo 20 años. ¿Qué edad tiene mi pareja?
Salida	63 años

Tabla 11. Ejemplo de pregunta que intenta resolver un problema matemático

Vaya. Obviamente, es una respuesta incorrecta. De hecho, los LLM suelen tener problemas con las tareas matemáticas y pueden dar respuestas incorrectas, incluso para una tarea tan sencilla como multiplicar dos números. Esto se debe a que se les entrena con grandes volúmenes de texto y las matemáticas pueden requerir un enfoque diferente. Así que veamos si los pasos intermedios de razonamiento mejoran el resultado.

<p><b>Pregunte a</b></p>	<p>Cuando tenía 3 , mi pareja tenía 3 veces mi edad. Ahora tengo 20 años. ¿Qué edad tiene mi pareja? Pensemos paso a paso.</p>
<p><b>Salida</b></p>	<ol style="list-style-type: none"> <li>1. Supongamos que mi edad actual es de "x" años.</li> <li>2. Cuando 3 años, mi edad era de 3 años.</li> <li>3. La edad de mi pafiner en ese momento era 3 veces mi edad, lo que significa que la edad de mi pafiner era <math>3 * 3 = 9</math> años.</li> <li>4. Ahora tengo 20 años, lo que significa que mi edad ha aumentado en <math>20 - 3 = 17</math> años desde que tenía 3 años.</li> <li>5. Dado que la edad de mi pafiner era de 9 años cuando yo tenía 3 años, y mi edad ha aumentado 17 años, la edad de mi pafiner también debe haber aumentado 17 años.</li> <li>6. Por lo tanto, la edad actual de mi pafiner es de <math>9 + 17 = 26</math> años.</li> </ol> <p>Mi pafiner tiene 26 .</p>

Tabla 12. Ejemplo de cadena de pensamiento

Muy bien, ahora la respuesta final es correcta. Esto se debe a que hemos dado instrucciones claras al LLM para que explique cada paso en lugar de limitarse a devolver una respuesta. Es interesante ver que el modelo suma los 17 años aumentados. En mi cabeza tomaría la diferencia de años entre mi pafiner y yo y los sumaría.  $(20 + (9 - 3))$ . Ayudemos al modelo a pensar un poco más como yo.

El cuadro 12 es un ejemplo de cadena de pensamiento de "disparo cero". La cadena de pensamiento puede ser muy poderosa cuando se combina con un solo disparo o pocos disparos, como se puede ver en la Tabla 13:

<p><b>Pregunte a</b></p>	<p>P: Cuando mi hermano 2 años, yo le doblaba la edad. Ahora tengo 40 años. ¿Qué edad tiene mi hermano? Pensemos paso a paso paso a paso.</p> <p>R: Cuando mi hermano tenía 2 años, yo tenía <math>2 * 2 = 4</math> años. Eso es una diferencia de edad de 2 años y yo soy mayor. Ahora tengo 40 , así que mi hermano tiene <math>40 - 2 = 38</math> años. La respuesta es 38.</p> <p>P: Cuando yo 3 años, mi pareja tenía 3 veces mi edad. Ahora, Tengo 20 . ¿Qué edad tiene mi pareja? Pensemos paso a paso.</p> <p>A:</p>
<p><b>Salida</b></p>	<p>Cuando yo tenía 3 años, mi pafiner tenía <math>3 * 3 = 9</math> años. Eso es una diferencia de edad de 6 años y mi pafiner es mayor. Ahora tengo 20 años, así que mi pafiner tiene <math>20 + 6 = 26</math> años. La respuesta es 26.</p>

Tabla 13. Ejemplo de cadena de pensamiento con un solo disparo

La cadena de pensamiento puede ser útil para varios casos de uso. Piense en la generación de código, para dividir la solicitud en unos pocos pasos y asignarlos a líneas específicas de . O para crear datos sintéticos cuando tienes algún tipo de semilla como *"El producto se llama XYZ, escribe una descripción guiando al modelo a través de las suposiciones que harías basándote en el producto dado el título"*. En general, cualquier tarea que pueda resolverse "hablando" es una buena candidata para una cadena de pensamiento. Si puedes explicar los pasos para resolver el problema, prueba con la cadena de pensamiento.

Consulte el cuaderno<sup>10</sup> alojado en el repositorio Github de GoogleCloudPlatform, en el que se explican con más detalle las indicaciones de CoT:

En la sección de mejores prácticas de este capítulo, aprenderemos algunas de las mejores prácticas específicas de la incitación a la cadena de pensamiento.

## Auto- coherencia

Aunque los modelos lingüísticos de gran tamaño han demostrado un éxito impresionante en diversas tareas de PNL, su capacidad de razonamiento se considera a menudo una limitación que no puede superarse únicamente aumentando el tamaño del modelo. Como hemos visto en la sección anterior sobre la cadena de pensamiento, se puede pedir al modelo que genere pasos de razonamiento como un ser humano que resuelve un problema. Sin embargo, CoT utiliza una estrategia simple de "descodificación codiciosa", lo que limita su eficacia. Autoconsistencia<sup>11</sup> combina el muestreo y la votación por mayoría para generar diversas vías de razonamiento y seleccionar la respuesta más coherente. Mejora la precisión y la coherencia de las respuestas generadas por los LLM.

La autoconsistencia da una pseudoprobabilidad de que una respuesta sea correcta, pero obviamente tiene altos costes.

Sigue los siguientes pasos:

1. Generación de diversas vías de razonamiento: El LLM recibe varias veces la misma pregunta. Un ajuste de temperatura alto anima al modelo a generar diferentes vías de razonamiento y perspectivas sobre el problema.
2. Extraiga la respuesta de cada respuesta generada.
3. Elija la respuesta más común.

Veamos un ejemplo de un sistema de clasificación de correos electrónicos, que clasifica un correo electrónico como IMPORTANTE o NO IMPORTANTE. Se enviará varias veces al LLM una cadena de pensamiento de disparo cero, para ver si las respuestas difieren después de cada envío. Fíjate en el tono amistoso, la elección de palabras y el sarcasmo que se ha utilizado en el correo electrónico. Todo esto podría engañar al LLM.

Pregunt  
e a

CORREO ELECTRÓNICO:

Hol

a,

He visto que utiliza Wordpress para su sitio web. Un gran sistema de gestión de contenidos de código abierto. Yo también lo he utilizado en el pasado. Viene con un montón de grandes plugins de usuario. Y es bastante fácil de configurar.

He observado un error en el formulario de contacto, que se produce al seleccionar el del nombre. Ver la captura de pantalla adjunta de mí introducir texto en el de nombre. Observe el cuadro de alerta JavaScript que inv0k3d.

Pero por lo demás es una gran página web. Disfruto . Siéntase libre de dejar el error en el sitio web, porque me da más cosas interesantes para leer.

Salud,

Harry el Hacker.

...

Clasifique el correo electrónico anterior como IMPORTANTE o NO IMPORTANTE. Pensemos paso a paso y expliquemos por qué.

**Continúa en la página siguiente...**

<p>Salida Aflemp 1</p>	<p><b>**Paso 1: Identificar el objetivo del correo electrónico.</b> El objetivo del correo electrónico es informar al destinatario de un error en el formulario de contacto de su sitio web. El remitente también menciona que le gusta leer el sitio web y que, de momento, deja el fallo en su sitio.</p> <p><b>**Paso 2: Evaluar el impacto potencial del fallo.</b> El fallo en el formulario de contacto podría permitir a un atacante ejecutar código JavaScript arbitrario en el sitio web. Esto podría llevar al atacante a tomar el control del sitio web, robar información sensible o desfigurar el sitio web.</p> <p><b>**Paso 3: Considera la credibilidad del remitente.</b> El remitente del correo electrónico afirma ser un hacker llamado Harry. Sin embargo, no hay forma de verificar la identidad del remitente. Es posible que el remitente sea un hacker legítimo que intenta ayudar al , pero también es posible que sea un actor malicioso que intenta engañar al destinatario para que haga clic en un enlace malicioso o descargue un archivo malicioso.</p> <p><b>**Conclusión: IMPORTANTE**</b> En función del impacto potencial del fallo y de la credibilidad del remitente, el correo electrónico debe clasificarse como <b>**IMPORTANTE**</b>. El destinatario debe tomar medidas inmediatas para corregir el fallo y proteger su sitio web de los ataques.</p>
<p>Salida Aflemp 2</p>	<p><b>**Explicación:**</b></p> <ol style="list-style-type: none"> <li><b>**Falta de urgencia:**</b> El correo electrónico no transmite ninguna sensación de urgencia o de que el tiempo apremia. Se inicia con un saludo informal y presenta la información en un tono práctico.</li> <li><b>**El fallo notificado en el formulario de contacto se describe como interesante pero no crítico.</b> El remitente del correo electrónico incluso sugiere que el propietario del sitio web puede dejar el fallo en su sitio.</li> <li><b>**Falta de impacto personal:**</b> El correo electrónico no indica ningún impacto directo o personal sobre el propietario del sitio web. Simplemente menciona que el remitente disfruta leyendo el sitio web.</li> </ol> <p style="text-align: right;"><b>Continúa en la página siguiente...</b></p>

<p>Salida Aflempt 2</p>	<p>4. <b>**Ausencia de solicitud de acción:**</b> El correo electrónico no solicita explícitamente ninguna acción por parte del propietario del sitio web. No pide que se solucione el fallo ni proporciona ninguna sugerencia específica de mejora.</p> <p>5. <b>**Intención del remitente:**</b> La intención del remitente parece estar más centrada en compartir una observación y expresar interés por el sitio web que en plantear una preocupación seria o solicitar ayuda.</p> <p><b>**Conclusión: SIN IMPORTANCIA**</b></p>
<p>Salida Aflempt 3</p>	<p><b>**Paso 1: Identificar el objetivo del correo electrónico.</b> El objetivo del correo electrónico es informar al destinatario de un error en el formulario de contacto de su sitio web. El remitente también menciona que le gusta leer el sitio web y que, de momento, deja el fallo en su sitio.</p> <p><b>**Paso 2: Evaluar el impacto potencial del fallo.</b> El fallo en el formulario de contacto podría permitir a un atacante ejecutar código JavaScript arbitrario en el sitio web. Esto podría llevar al atacante a tomar el control del sitio web, robar información sensible o desfigurar el sitio web.</p> <p><b>**Paso 3: Considera la credibilidad del remitente.</b> El remitente del correo electrónico afirma ser un hacker llamado Harry. Sin embargo, no hay forma de verificar la identidad del remitente. Es posible que el remitente sea un hacker legítimo que está tratando de ayudar al destinatario, pero también es posible que el remitente sea un actor malicioso que está tratando de explotar el fallo para su propio beneficio.</p> <p><b>**Conclusión:**</b> Basándonos en los factores anteriores, el correo electrónico debe clasificarse como <b>**IMPORTANTE**</b>. El error en el formulario de contacto supone un grave riesgo para la seguridad y se desconoce la credibilidad del remitente. El destinatario debe tomar medidas inmediatas para corregir el fallo y</p>

Tabla 14. Ejemplo de ayuda a la coherencia

Puede utilizar la consulta anterior e intentar ver si devuelve una clasificación coherente. Dependiendo del modelo que esté utilizando y de la configuración de la temperatura, podría devolver "IMPORTANTE" o "NO IMPORTANTE".

Al generar muchas Cadenas de Pensamientos y tomar la respuesta más frecuente ("IMPORTANTE"), podemos obtener una respuesta correcta más consistente del LLM.

Este ejemplo muestra cómo se puede utilizar el estímulo de autoconsistencia para mejorar la precisión de la respuesta de un LLM al considerar múltiples perspectivas y seleccionar la respuesta más coherente.

## Árbol de los pensamientos (ToT)

Ahora que estamos familiarizados con la cadena de pensamiento y el estímulo de autoconsistencia, repasemos el Árbol de Pensamientos (TdP)<sup>12</sup>. Generaliza el concepto de estímulo TdP porque permite a los LLM explorar múltiples vías de razonamiento simultáneamente, en lugar de seguir una única cadena de pensamiento lineal. Esto se representa en la Figura 1.

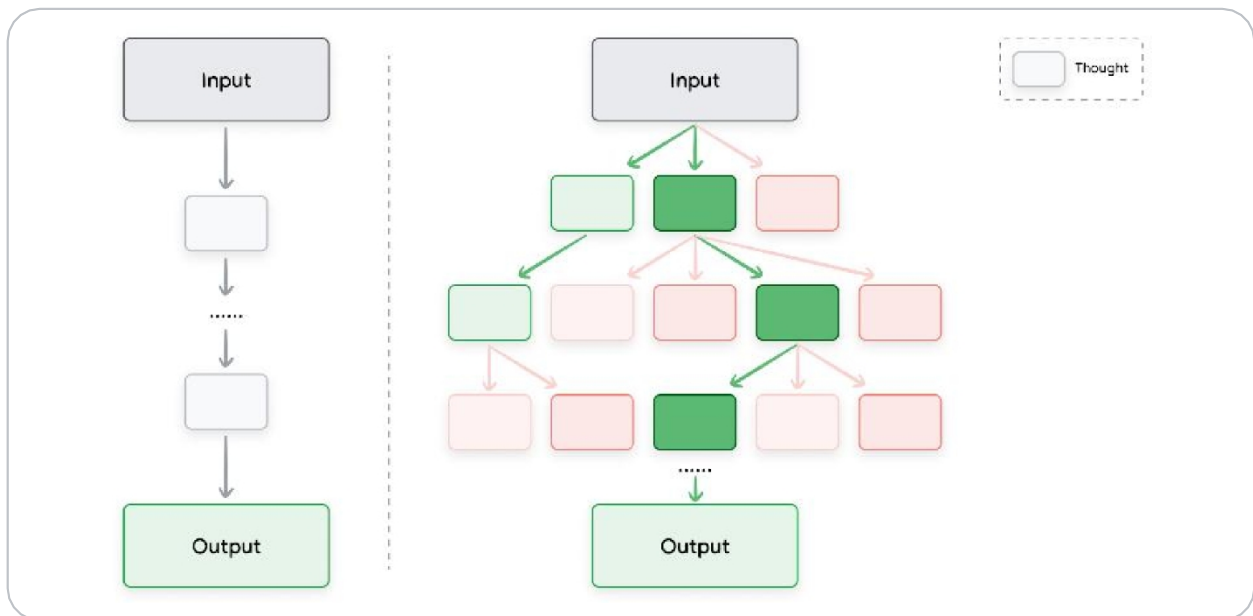


Figura 1. Visualización de la cadena de pensamientos en el leh versus. Árbol de pensamientos que incita a el derecho

Este enfoque hace que ToT sea especialmente adecuado para tareas complejas que requieren exploración. Funciona manteniendo un árbol de pensamientos, donde cada pensamiento representa una secuencia lingüística coherente que sirve como paso intermedio hacia la resolución de un problema. El modelo puede explorar distintas vías de razonamiento partiendo de distintos nodos del árbol.

Hay un gran cuaderno, que entra en un poco más de detalle mostrando El Árbol del Pensamiento (ToT) que se basa en el documento 'Modelo de Lenguaje Grande Guiado Árbol de Pensamiento'.<sup>9</sup>

## ReAct (reason & act)

Reason and act (ReAct) [10]<sup>13</sup> prompting es un paradigma para permitir a los LLMs resolver tareas complejas utilizando razonamiento en lenguaje natural combinado con herramientas externas (búsqueda, intérprete de código, etc.) que permiten al LLM realizar acciones cefiain, como interactuar con APIs externas para recuperar información, lo que supone un primer paso hacia el modelado de agentes.

ReAct imita la forma en que los humanos actuamos en el mundo real, ya que razonamos verbalmente y podemos emprender acciones para obtener información. ReAct obtiene buenos resultados frente a otros enfoques de ingeniería rápida en diversos ámbitos.

ReAct funciona combinando el razonamiento y la acción en un bucle de pensamiento-acción. El LLM primero razona sobre el problema y genera un plan de acción. A continuación, ejecuta las acciones del plan y observa los resultados. A continuación, el LLM utiliza las observaciones para actualizar su razonamiento y generar un nuevo plan de acción. Este proceso continúa hasta que el LLM alcanza una solución al problema.

Para ver esto en acción, necesitas escribir algo de código. En el fragmento de código 1 estoy utilizando el framework langchain para Python, junto con VefiexAI (google-cloud-aiplatform) y los paquetes `pip google-search-results`.

Para ejecutar este ejemplo debe crear una clave SerpAPI (gratuita) desde <https://serpapi.com/manage-api-key> y establecer una variable de entorno `SERPAPI_API_KEY`.

A continuación vamos a escribir un poco de código Python, con la tarea para el LLM para averiguar: *Cuántos niños tienen un padre famoso que toca en la banda Metallica.*

### Python

```
from langchain.agents import carga_herramientas
from langchain.agents import inicializar_agente
from langchain.agents import Tipo_agente
from langchain.llms import VertexAI

prompt= "¿Cuántos hijos tienen los miembros de la banda Metallica?"

llm = VertexAI(temperatura= 0.1)
tools= load_tools(["serpapi"], llm= llm)

agent = initialize_agent(tools, llm,
agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION, verbose= True)
agent.run(prompt)
```

Fragmento 1. Creación de un agente ReAct con LangChain y VefiexAI

El fragmento de código 2 muestra el resultado. Observa que ReAct realiza una cadena de cinco búsquedas. De hecho, el LLM está raspando los resultados de búsqueda de Google para averiguar los nombres de las bandas. A continuación, enumera los resultados como observaciones y encadena el pensamiento para la siguiente búsqueda.

```

> Entrando en la nueva cadena AgentExecutor...
Metallica tiene 4 miembros.
Acción: Buscar en
Entrada de acción: ¿Cuántos hijos tiene James Hetfield?
Observación: tres hijos
Pensamiento: 1/4 miembros de la banda Metallica tienen 3
hijos Acción: Buscar en
Entrada de acción: ¿Cuántos hijos tiene Lars Ulrich?
Observación: 3
Pensamiento: 2/4 miembros de la banda Metallica tienen 6
hijos Acción: Buscar en
Entrada de acción: ¿Cuántos hijos tiene Kirk Hammett?
Observación: Hammett está casado con su segunda esposa, Lani, desde 1998.
Tienen dos hijos, Angel (nacido el 29 de septiembre de 2006) y Vincenzo
(nacido el 28 de junio de 2008).
Pensamiento: 3/4 miembros de la banda Metallica tienen 8
hijos Acción: Buscar en
Entrada de acción: ¿Cuántos hijos tiene Robert Trujillo?
Observación: 2
Pensamiento: 4/4 Los miembros de la banda Metallica tienen
10 hijos Respuesta final: 10

```

#### Fragmento 2. Ejecución del agente ReAct en el terminal

El fragmento de código 2 averigua que la banda Metallica tiene cuatro miembros. Luego busca en cada miembro de la banda para pedir el total de hijos y suma el total. Por último, devuelve el número total de hijos como respuesta final.

En la práctica, el envío de mensajes ReAct requiere comprender que hay que reenviar continuamente los mensajes/respuestas anteriores (y recortar el contenido extra generado), así como configurar el modelo con ejemplos/instrucciones apropiados. Consulte el cuaderno<sup>14</sup> alojado en el repositorio Github de GoogleCloudPlatform, que entra en un poco más de detalle mostrando las entradas y salidas reales de LLM con un ejemplo más elaborado.

## Aviso automático Ingeniería

Llegados a este punto, puede que te des cuenta de que escribir un aviso puede ser complejo. ¿No estaría bien automatizar esto (escribir un aviso para escribir avisos)? Bueno, existe un método: Automatic Prompt Engineering (APE). Este método<sup>15</sup> no sólo alivia la necesidad de la intervención humana, sino que también mejora el rendimiento del modelo en diversas tareas.

Le pedirá a un modelo que genere más indicaciones. Evalúelas, posiblemente modifique las buenas. Y repetir.

Por ejemplo, puede utilizar la ingeniería de instrucciones automáticas que le ayudará a entrenar un chatbot para una tienda online de productos t-shifi. Queremos averiguar todas las diversas formas en que los clientes podrían frase su pedido para la compra de una mercancía banda t-shifi.

1. Escribe el prompt que generará las variantes de salida. En este ejemplo, estoy usando gemini-pro para generar 10 instrucciones. Ver Tabla 15:

<p><b>Pregunte a</b></p>	<p>Tenemos una tienda online de camisetas con merchandising de grupos de música, y para entrenar a un chatbot necesitamos varias formas de hacer un pedido: "Una camiseta de Metallica talla S". Generar 10 variantes, con la misma semántica pero manteniendo el mismo significado.</p>
<p><b>Salida</b></p>	<ol style="list-style-type: none"> <li>1. Me gustaría comprar un Metallica t-shifi en talla pequeña.</li> <li>2. ¿Puedo pedir un t-shifi de Metallica de tamaño pequeño?</li> <li>3. Me gustaría comprar una Metallica t-shifi en pequeño.</li> <li>4. Un shifi de Metallica, talla pequeña, por favor.</li> <li>5. Quiero un Metallica t-shifi en talla pequeña.</li> <li>6. Me gustaría hacer un pedido de un Metallica t-shifi en pequeño.</li> <li>7. Pequeño Metallica t-shifi, uno por favor.</li> <li>8. Busco una Metallica t-shifi en pequeño.</li> <li>9. ¿Puedes conseguirme un Metallica t-shifi en pequeño?</li> <li>10. Un Metallica t-shifi pequeño, por favor.</li> </ol>

Tabla 15. Utilizar un LLM para generar avisos

2. Evalúe todos los candidatos a la instrucción puntuando a los candidatos en función de una métrica elegida, Por ejemplo, puede utilizar BLEU (Bilingual Evaluation Understudy) o ROUGE (Recall-Oriented Understudy for Gisting Evaluation).
3. Seleccione el candidato a instrucción con la puntuación de evaluación más alta. Este candidato será la instrucción final que podrá utilizar en su aplicación de software o chatbot. También puede modificar la instrucción seleccionada y evaluarla de nuevo.

## Código

Gemini se centra principalmente en instrucciones basadas en texto, que también incluyen instrucciones de escritura para devolver código. Vayamos a Vefiex AI Studio y probemos estas instrucciones para ver algunos ejemplos de codificación.

### Sugerencias para escribir código

Gemini también puede ser un desarrollador y ayudarte a escribir código en cualquier lenguaje de programación que elijas. Como desarrollador, esto puede ayudarte a acelerar el proceso de escritura de código.

Imagine una carpeta en su ordenador con cientos de archivos que necesita renombrar. Renombrar cada archivo te llevaría mucho tiempo. Usted sabe un poco de Bash, y podría escribir un script para automatizar esto, pero eso también puede tomar un tiempo. Así que vamos a escribir un prompt. Puedes escribir el mensaje en el chatbot Gemini, o si estás más preocupado sobre confidencialidad, puedes escribir estas indicaciones dentro de tu cuenta de Google Cloud y abrir Vefiex AI Studio. La ventaja de Vefiex AI Studio es que puedes configurar la temperatura, etc.

Nombre	1_bash_rename_files_in_folder		
Objetivo	Escribe un prompt para escribir código en Bash para renombrar archivos en un		carpeta .
Modelo	gemini-pro		
Temperatura	0.1	Límite de fichas	1024
Top-K	N/A	Top-P	1
Pregunta	Escribe un fragmento de código en Bash, que pregunte por el nombre de una carpeta. A continuación, toma el contenido de la carpeta y cambia el nombre de todos archivos que contiene anteponiendo el nombre borrador al nombre del archivo.		
Salida	<pre> ```bash #!/bin/bash  # Pide el nombre de la carpeta echo "Introduce el nombre de la carpeta: " read nombre_carpeta  # Comprueba si la carpeta existe if [ ! -d "\$nombre_carpeta" ]; then echo "La carpeta no existe" exit 1 fi  # Obtener la lista de archivos de la carpeta files=( "\$nombre_carpeta"/* )  # Renombra cada archivo anteponiendo "borrador" al nombre del archivo para archivo en "\${archivos[@]}"; do nuevo_nombre_archivo="borrador_\${nombre_base "\$archivo}" mv "\$archivo" "\$nuevo_nombre_archivo" done  echo "Archivos renombrados con éxito". ``` </pre>		

Tabla 16. Utilización de Gemini para programar código

A mí me parece un buen código, ¡incluso está documentado! Sin embargo, como los LLM no pueden razonar, y repiten los datos de entrenamiento, es esencial leer y probar primero tu código.

El momento que todos esperamos, ¿funciona de verdad?

Vamos a probarlo primero con una carpeta de prueba con sólo unos pocos archivos en ella, que tendrá que ser renombrada de `filename.txt` a `draft_filename.txt`.

1. Copia la salida de la Tabla 16 (sin la envoltura de texto `` **bash** ``), y pégala en un nuevo archivo llamado: "**renombrar\_archivos.sh**".
2. Abra una ventana de terminal y escriba: **. rename\_files.sh**. Le pedirá que introduzca un nombre de carpeta, por ejemplo **test**. y pulsa enter.
3. El script parece funcionar correctamente. Verá el mensaje: **Archivos renombrados con éxito**. Cuando mire en la carpeta de prueba, observe que todos los archivos se renombraron perfectamente a **nombre\_borrador.txt**.

¡Funcionó!

## Sugerencias para explicar el código

Como desarrollador, cuando trabajas en equipo tienes que leer el código de los demás. Gemini también puede ayudarte con esto. Tomemos la salida de código de la Tabla 16, eliminemos los comentarios y pidamos al gran modelo de lenguaje que explique lo que está sucediendo, Ver Tabla 17:

Nombre	1_bash_explica_el_código		
Objetivo	Escribe un mensaje para explicar el código Bash		
Modelo	gemini-pro		
Temperatura	0.1	Límite de fichas	1024
Top-K	N/A	Top-P	1
Pregunte a	<p>Explícame el siguiente código Bash:</p> <pre> ''' #!/bin/bash  echo "Introduzca el nombre de la carpeta: " read nombre_carpeta  if [ ! -d "\$nombre_carpeta" ]; then echo "La carpeta no existe" exit 1 fi  ficheros=( "\$nombre_carpeta"/* ) for fichero in "\${ficheros[@]}; do nuevo_nombre_archivo="borrador_\$(nombre_base "\$archivo)" mv "\$archivo" "\$nuevo_nombre_archivo" hecho </pre>		
Salida	<pre> echo "Archivos renombrados con éxito". '''  Este script Bash realiza las siguientes tareas:  1. <b>**Entrada de usuario:</b> - Pide al usuario que introduzca un nombre de carpeta utilizando los comandos `echo` y `read`. El nombre de carpeta introducido se almacena en la variable `folder_name`.  2. <b>**Comprobación de existencia de carpetas**:</b> - Comprueba si la carpeta con el nombre especificado existe utilizando la condición `[ ! -d "\$nombre_carpeta" ]`. Si la carpeta no existe, muestra un mensaje de error ("Folder does not exist.") y sale del script con un código de salida de 1. </pre>		

Continúa en la página siguiente...

Salida	<p>3. <b>**Lista de archivos:</b></p> <ul style="list-style-type: none"> <li>- Lista todos los archivos dentro de la carpeta especificada usando el comando <code>files=("\$nombre_carpeta"/*)`</code>. Esto crea una matriz llamada <code>files`</code> que contiene las rutas de todos los archivos de la carpeta.</li> </ul> <p>4. <b>**Renombramiento de archivos**:</b></p> <ul style="list-style-type: none"> <li>- Recorre cada archivo de la matriz <code>files`</code> mediante un bucle <code>for`</code>.</li> <li>- Para cada fichero, genera un nuevo nombre de fichero utilizando el <code>nuevo_fichero_nombre="borrador_\$(nombre base "\$fichero")`</code>. Este nuevo nombre de archivo se crea anteponiendo al nombre del archivo original la cadena "borrador_".</li> <li>- A continuación, cambia el nombre del archivo mediante el comando <code>mv "\$archivo" "\$nuevo_nombre_archivo"</code>, que mueve el archivo al nuevo nombre de archivo.</li> </ul> <p>5. <b>**Mensaje de éxito:</b></p> <ul style="list-style-type: none"> <li>- Una vez renombrados todos los archivos, muestra un mensaje de éxito ("Files renamed successfully.") para indicar que la operación de renombrado se ha realizado correctamente</li> </ul>
--------	---

Tabla 17. Una indicación para explicar el código

## Indicaciones para traducir el código

El código bash de la Tabla 16 parece funcionar bien. Sin embargo este script podría ser realmente reutilizable cuando me pidiera los nombres de los archivos, lo ideal sería que funcionara como una aplicación independiente con una interfaz de usuario. Como punto de partida, Python sería un mejor lenguaje para una aplicación (web) que Bash. Los LLM pueden ayudar a traducir código de un lenguaje a otro. Véase el ejemplo de la Tabla 18:

<b>Nombre</b>	1_python_rename_files		
<b>Objetivo</b>	Escribir un prompt para traducir código Bash a		
<b>Modelo</b>	Python gemini-pro		
<b>Temperatura</b>	0.1	<b>Límite de fichas</b>	1024
<b>Top-K Prompt</b>	N/A	<b>Top-P</b>	1
<b>Salida</b>	<p>Traduzca el siguiente código Bash a un fragmento de Python.</p> <pre> ```bash #!/bin/bash  echo "Introduzca el nombre de la carpeta: " read nombre_carpeta  if [ ! -d "\$nombre_carpeta" ]; then echo "La carpeta no existe" exit 1 fi  ficheros=( "\$nombre_carpeta"/* ) for fichero in "\${ficheros[@]}"; do nuevo_nombre_archivo="borrador_\$(nombre_base "\$archivo)" mv "\$archivo" "\$nuevo_nombre_archivo" hecho </pre>		
	<pre> echo "Archivos renombrados con éxito". ```  ```python import os import shutil  # Obtener el nombre de la carpeta del usuario nombre_carpeta= input("Introduzca el nombre de la carpeta: ")  # Comprueba si la carpeta existe if not os.path.isdir(nombre_carpeta): print("La carpeta no existe.") salir(1) </pre>		

**Continúa en la página siguiente...**

<b>Salida</b>	<pre> # Obtener una lista de todos los archivos de la carpeta archivos = os.listdir(nombre_carpeta)  # Iterar sobre los archivos para archivo en archivos:     # Crear un nuevo nombre de archivo con el prefijo     "borrador_" nuevo_nombre_archivo = f     "borrador_{archivo}"  # Mueve el archivo al nuevo nombre shutil.move(os.path.join(nombre_carpeta, archivo), os.path.join(nombre_carpeta, nuevo_nombre_archivo))  # Imprime un mensaje de éxito print("Archivos renombrados con éxito.") ` </pre>
---------------	--

Tabla 18. Instrucciones para traducir código de Bash a Python

Lee y revisa el código. Copia la salida del prompt y pégala en un nuevo archivo: **file\_renamer.py**. Prueba el código abriendo una ventana de Terminal, y ejecuta el siguiente comando **python file\_renamer.py**.

**NOTA:** Cuando se te pida código (Python) en Language Studio en Vefiex AI, tendrás que hacer clic en el botón 'Markdown'. De lo contrario, recibirá texto sin formato que carece de la sangría adecuada de las líneas, que es impofiant para la ejecución de código Python.

## Sugerencias para depurar y revisar el código

Escribamos manualmente algunos cambios en el código de la Tabla 18. Debería preguntar al usuario por el prefijo del nombre de fichero, y escribir este prefijo en . Vea el código de ejemplo en el Snippet \*3, pero qué fastidio. ¡Ahora devuelve errores Python!

**Python**

```

import os
import shutil

nombre_carpeta= input("Introduzca el nombre de la carpeta: ")
prefix= input("Introduzca la cadena que desea añadir al nombre de
archivo: ") text = toUpperCase(prefix)

if not os.path.isdir(nombre_carpeta):
    print("La carpeta no existe.") exit(1)

archivos= os.listdir(nombre_carpeta)

para archivo en archivos:
    nuevo_nombre_archivo= f"{texto}_{archivo}"

shutil.move(os.path.join(nombre_carpeta,
archivo), os.path.joi(nombre_carpeta,
nuevo_nombre_archivo))

print("Archivos renombrados con éxito.")

```

Fragmento 3. Un script de Python roto

¡Uf! Eso parece un bicho:

```

El siguiente código Python da un error:

Traceback (llamada más reciente):
  File "/Users/leeboonstra/Documents/test_folder/rename_files.py", line 7, in
<module>.
    texto= toUpperCase(prefijo)
NameError: el nombre 'toUpperCase' no está definido

```

Fragmento 4. He roto el código Python

Veamos si podemos pedir al gran modelo lingüístico que depure y revise el código. Echa un vistazo a la Tabla 19:

Nombre	1_python_debug_code		
Objetivo	Escribe un prompt para depurar y revisar código Python.		
Modelo	gemini-pro		
Temperatura	0.1	Límite de fichas	1024
Top-K	N/A	Top-P	1
Pregunte a	<p>El siguiente código Python da un error:</p> <pre> Traceback (llamada más reciente):   File "/ Users/leeboonstra/Documents/test_folder/rename_files.py", line 7, in &lt;module&gt;.     texto= toUpperCase(prefijo) NameError: el nombre 'toUpperCase' no está definido  Depurar lo que está mal y explicar cómo puedo mejorar el .  ```python import os import shutil  # Obtener el nombre de la carpeta del usuario nombre_carpeta= input("Introduzca el nombre de la carpeta: ") prefix= input("Introduzca la cadena que desea añadir al nombre de archivo: ") text = toUpperCase(prefix)  # Comprueba si la carpeta existe if not os.path.isdir(nombre_carpeta):     print("La carpeta no existe.") exit(1)  # Obtener una lista de todos los archivos de la carpeta archivos= os.listdir(nombre_carpeta)  # Iterar sobre los archivos para archivo en archivos:                     </pre> <p style="text-align: right;"><b>Continúa en la página siguiente...</b></p>		

<p><b>Pregunte a</b></p>	<pre># Crea un nuevo nombre de archivo con el prefijo "borrador_" nuevo_nombre_archivo= f"{texto}_{archivo}"  # Mueve el archivo al nuevo nombre shutil.move(os.path.join(nombre_carpeta, archivo), os.path.join(nombre_carpeta, nuevo_nombre_archivo))  # Imprime un mensaje de éxito print("Archivos renombrados con éxito.") ...</pre>
<p><b>Salida</b></p>	<p>El código llama a la función `toUpperCase` para convertir `prefix` a mayúsculas, pero esa función no está definida. Para solucionar este problema, puede utilizar el método `upper()` de la clase string que convierte una cadena dada a mayúsculas. El código modificado se muestra a continuación:</p> <pre>```python import os import shutil  # Obtener el nombre de la carpeta del usuario nombre_carpeta= input("Introduzca el nombre de la carpeta: ") prefix= input("Introduzca la cadena que desea añadir al nombre de archivo: ") text = prefix.upper()  # Comprueba si la carpeta existe if not os.path.isdir(nombre_carpeta):     print("La carpeta no existe.")     exit(1)  # Obtener una lista de todos los archivos de la carpeta archivos= os.listdir(nombre_carpeta)  # Iterar sobre los archivos para archivo en archivos:     # Crear un nuevo nombre de archivo con el prefijo "borrador_"     nuevo_nombre_archivo= f"{texto}_{archivo}"  # Mueve el archivo al nuevo nombre shutil.move(os.path.join(nombre_carpeta, archivo),</pre> <p style="text-align: right;"><b>Continúa en la página siguiente...</b></p>

Salida	<pre>os.path.join(nombre_carpeta, nuevo_nombre_archivo))  # Imprime un mensaje de éxito print("Archivos renombrados con éxito.") '''</pre> <p>Además, se pueden introducir algunas mejoras en el código:</p> <ol style="list-style-type: none"> <li>1. La extensión de los nuevos nombres de archivo no se conserva. Es mejor extraer la extensión del nombre de archivo original y añadirla al nuevo nombre de archivo.</li> <li>2. El código asume que no hay espacios en el nombre de la carpeta. Sin embargo, es una buena práctica tratar los espacios con elegancia sustituyéndolos por guiones bajos o guiones.</li> <li>3. Utilizar la sintaxis de cadena `f` para la interpolación de cadenas es más legible y conciso que el operador tradicional `+`.</li> <li>4. El código no maneja los errores que puedan ocurrir durante el proceso de renombrado. Sería mejor envolver la llamada a `shutil.move` en un bloque `try...except` para capturar cualquier error potencial.</li> </ol> <p>Aquí está el código mejorado con estas sugerencias:</p> <pre>```python import os import shutil # Obtener el nombre de la carpeta del usuario nombre_carpeta= input("Introduzca el nombre de la carpeta: ") prefix= input("Introduzca la cadena a añadir al nombre del archivo: ") # Compruebe si la carpeta existe if not os.path.isdir(nombre_carpeta):     print("La carpeta no existe.")     exit(1) # Obtener una lista de todos los archivos de la carpeta archivos= os.listdir(nombre_carpeta) # Iterar sobre los archivos para archivo en archivos:</pre> <p style="text-align: right;"><b>Continúa en la página siguiente...</b></p>
--------	---

Salida	<pre> # Obtener la extensión del archivo ext= os.path.splitext(archivo)[1]  # Crear un nuevo nombre de archivo con el prefijo y la extensión de archivo new_filename = f"{prefix}_{os.path.splitext(file)[0]}{ext}"  # Construye las rutas completas para los archivos original y nuevo old_path= os.path.join(nombre_carpeta, archivo) new_path= os.path.join(nombre_carpeta, nuevo_nombre_archivo)  # Intenta mover el archivo intenta:     shutil.move(ruta_antigua, ruta_nueva)  excepto Excepción como e:     print(f "Se ha producido un error al mover el archivo {archivo}: {e}")  # Imprime un mensaje de éxito print("Archivos renombrados con éxito.") '''  Estas mejoras hacen que el código sea más robusto y flexible, a la vez que más fácil de leer y entender... (La respuesta se ha truncado porque ha alcanzado el límite de tokens. Intenta aumentar el límite de tokens si necesitas una respuesta más larga). </pre>
--------	--

Tabla 19. Un indicador para depurar y revisar código Python

Eso es genial. No sólo me dijo cómo resolver el problema, sino que también averiguó que mi código tenía más fallos y cómo resolverlos también. El último párrafo del prompt daba sugerencias para mejorar el código en general.

## ¿Qué hay de la indicación multimodal ?

La petición de código sigue utilizando el mismo modelo de lenguaje de gran tamaño. La orientación multimodal es una cuestión aparte, y se refiere a una técnica en la que se utilizan varios formatos de entrada para guiar un modelo de lenguaje de gran tamaño, en lugar de basarse únicamente en el texto. Esto puede incluir combinaciones de texto, imágenes, audio, código o incluso otros formatos, dependiendo de las capacidades del modelo y de la tarea en cuestión.

## Mejores prácticas

Para encontrar el mensaje adecuado hay que jugar. Language Studio en Vefiex AI es un lugar perfecto para jugar con sus mensajes, con la capacidad de probar contra los distintos modelos.

Utiliza las siguientes buenas prácticas para convertirte en un profesional de la ingeniería rápida.

### Proporcione ejemplos

La mejor práctica más importante consiste en proporcionar ejemplos (de una o varias tomas) dentro de una pregunta. Esto es muy eficaz porque actúa como una poderosa herramienta de enseñanza. Estos ejemplos muestran resultados deseados o respuestas similares, lo que permite al modelo aprender de ellos.

y adaptar su propia generación en consecuencia. Es como darle al modelo un punto de referencia o un objetivo al que apuntar, mejorando la precisión, el estilo y el tono de su respuesta para que se ajuste mejor a tus expectativas.

## Diseñar con sencillez

Las instrucciones deben ser concisas, claras y fáciles de entender, tanto para ti como para el modelo. Como regla general, si ya es confuso para usted, es probable que también lo sea para el modelo. Intente no utilizar un lenguaje complejo y no proporcione información innecesaria.

Ejemplos:

ANTES:

Estoy de visita en Nueva York en este momento, y me gustaría saber más acerca de grandes lugares. Voy con dos niños de 3 años. ¿Dónde deberíamos ir durante nuestras vacaciones?

DESPUÉS DE LA REESCRITURA:

Actuar como guía de viajes para turistas. Describe lugares estupendos para visitar en Nueva York Manhattan con un niño de 3 años.

Intenta utilizar verbos que describan la acción. Aquí tienes una serie de ejemplos:

**Actuar, Analizar, Categorizar, Clasificar, Contrastar, Comparar, Crear, Describir, Definir, Evaluar, Extraer, Encontrar, Generar, Identificar, Listar, Medir, Organizar, Parsear, Elegir, Predecir, Proporcionar, Clasificar, Recomendar, Devolver, Recuperar, Reescribir, Seleccionar, Mostrar, Sofi, Resumir, Traducir, Escribir.**

## Especifique el resultado

Sea específico sobre el resultado deseado. Una instrucción concisa puede no guiar lo suficiente al LLM o ser demasiado genérica. Proporcionar detalles específicos en la instrucción (a través del sistema o del contexto) puede ayudar al modelo a centrarse en lo que es relevante, mejorando la precisión general.

Ejemplos:

DO:

Elabore una entrada de blog de 3 párrafos sobre las 5 mejores consolas de videojuegos. La entrada del blog debe ser informativa y atractiva, y debe estar escrita en un estilo conversacional.

NO LO HAGAS:

Generar una entrada de blog sobre consolas de videojuegos.

## Utilizar instrucciones en lugar de restricciones

Las instrucciones y restricciones se utilizan en los avisos para guiar la salida de un LLM.

- Una **instrucción** proporciona instrucciones explícitas sobre el formato, el estilo o el contenido deseados de la respuesta. Orienta al modelo sobre lo que éste debe hacer o producir.
- Una **restricción** es un conjunto de limitaciones o límites a la respuesta. Limita lo que el modelo no debe hacer o evitar.

Cada vez hay más estudios que sugieren que centrarse en las instrucciones positivas puede ser más eficaz que depender en gran medida de las restricciones. Este enfoque coincide con la preferencia humana por las instrucciones positivas frente a las listas de lo que no se debe hacer.

Las instrucciones comunican directamente el resultado deseado, mientras que las restricciones pueden dejar al modelo adivinando lo que está permitido. Las instrucciones dan flexibilidad y fomentan la creatividad dentro de los límites definidos, mientras que las restricciones pueden limitar el potencial del modelo. Además, una lista de restricciones puede chocar entre sí.

Las restricciones siguen siendo valiosas, pero en situaciones específicas. Para evitar que el modelo genere contenidos perjudiciales o sesgados o cuando se necesita un formato o estilo de salida estricto.

Si es posible, utilice instrucciones positivas: en lugar de decirle al modelo lo que no debe hacer, dígame lo que sí debe hacer. Esto puede evitar confusiones y mejorar la precisión del resultado.

**DO:**

Elabore una entrada de blog de 1 párrafo sobre las 5 principales consolas de videojuegos. Habla solo de la consola, la empresa que la fabricó, el año y las ventas totales.

**NO LO HAGAS:**

Elabore una entrada de blog de 1 párrafo sobre las 5 mejores consolas de videojuegos. No enumere nombres de videojuegos.

Como práctica recomendada, priorice las instrucciones, establezca claramente lo que quiere que haga el modelo y utilice restricciones sólo cuando sea necesario por motivos de seguridad, claridad o requisitos específicos. Experimenta e itera para probar diferentes combinaciones de instrucciones y restricciones hasta encontrar lo que mejor funciona para tus tareas específicas, y documéntalo.

## Controla la longitud máxima de las fichas

Para controlar la longitud de una respuesta LLM generada, puede establecer un límite máximo de tokens en la configuración o solicitar explícitamente una longitud específica en su consulta. Por ejemplo:

*"Explica la física cuántica en un mensaje de la longitud de un tuit".*

## Utilizar variables en los mensajes

Para reutilizar los avisos y hacerlos más dinámicos, utilice variables en el aviso, que pueden cambiarse para diferentes entradas. Por ejemplo, como se muestra en la Tabla 20, un prompt que da datos sobre una ciudad. En lugar de codificar el nombre de la ciudad en el prompt, utilice una variable. Las variables pueden ahorrarle tiempo y esfuerzo, ya que le permiten evitar repetir lo mismo. Si usted necesita utilizar la misma información en varias solicitudes, puede almacenarla en una variable y luego hacer referencia a esa variable en cada solicitud. Esto tiene mucho sentido al integrar avisos en sus propias aplicaciones.

Pregunte a	<p>VARIABLES</p> <pre>{ciudad} = "Amsterdam"</pre> <p>PROMPT</p> <p>Usted es un guía de viajes. Cuéntame algo sobre la : {ciudad}</p>
Salida	<p>Ámsterdam es una hermosa ciudad llena de canales, puentes y calles estrechas. Es un gran lugar para visitar por su rica historia, cultura y vida nocturna.</p>

Tabla 20. Uso de variables en avisos

## Experimentar con formatos de entrada y estilos de escritura

Diferentes modelos, configuraciones de modelos, formatos de pregunta, opciones de palabras y envíos pueden dar resultados diferentes. Por lo tanto, es importante experimentar con atributos como el estilo, la elección de palabras y el tipo de pregunta (cero disparos, pocos disparos, pregunta del sistema).

Por ejemplo, una pregunta con el objetivo de generar un texto sobre la revolucionaria consola de videojuegos Sega Dreamcast puede formularse como **una pregunta**, una **afirmación** o una **instrucción**, lo que da lugar a diferentes resultados:

- **Pregunta:** ¿Qué era la Sega Dreamcast y por qué fue una consola tan revolucionaria?
- **Declaración:** La Sega Dreamcast fue una videoconsola de sexta generación lanzada por Sega en 1999. Se...
- **Instrucciones:** Escribe un único párrafo que describa la consola Sega Dreamcast y explique por qué fue tan revolucionaria.

## Para realizar tareas de clasificación con pocos disparos, mezcle las clases

En general, el orden de los ejemplos de pocos disparos no debería importar mucho. Sin embargo, al realizar tareas de clasificación, asegúrese de mezclar las posibles clases de respuesta en ejemplos de pocos disparos. Esto se debe a que, de lo contrario, podría ajustarse en exceso al orden específico de los ejemplos. Al mezclar las posibles clases de respuesta, puede asegurarse de que el modelo está aprendiendo a identificar las características clave de cada clase, en lugar de simplemente memorizar el orden de los ejemplos. De este modo, se obtendrá un rendimiento más sólido y generalizable con datos desconocidos.

Una buena regla general es stafi con 6 pocos ejemplos de disparo y stafi probando la precisión a partir de ahí.

## **Adaptarse a las actualizaciones de los modelos**

Es importante que esté al tanto de los cambios en la arquitectura del modelo, los datos añadidos y las capacidades. Pruebe las nuevas versiones del modelo y ajuste sus avisos para aprovechar mejor las nuevas características del modelo. Herramientas como Vefiex AI Studio son ideales para almacenar, probar y documentar las distintas versiones de sus instrucciones.

## **Experimente con los formatos de salida**

Además del formato de entrada, considere la posibilidad de experimentar con el formato de salida. Para tareas no creativas como la extracción, selección, análisis sintáctico, ordenación, clasificación o categorización de datos, intente que la salida se devuelva en un formato estructurado como JSON o XML.

Hay algunos beneficios en devolver objetos JSON desde un prompt que extrae datos. En una aplicación del mundo real no necesito crear manualmente este formato JSON, ya puedo devolver los datos en un orden suavizado (muy útil cuando se trabaja con objetos datetime), pero lo más impofiantly, pidiendo un formato JSON obliga al modelo para crear una estructura y limitar las alucinaciones.

En resumen, ventajas de utilizar JSON para su salida:

- Devuelve siempre el mismo estilo
- Concéntrese en los datos que desea recibir

- Menos posibilidades de alucinaciones
- Que sea consciente de las relaciones
- Se obtienen tipos de datos
- Puedes sofi

La tabla 4 de la sección de avisos de pocos disparos muestra un ejemplo de cómo devolver una salida estructurada.

## Reparación JSON

Aunque devolver datos en formato JSON ofrece numerosas ventajas, no está exento de inconvenientes. La naturaleza estructurada de JSON, aunque beneficiosa para el análisis sintáctico y el uso en aplicaciones, requiere un número significativamente mayor de tokens que el texto sin formato, lo que conlleva un aumento del tiempo de procesamiento y de los costes. Además, la verbosidad de JSON puede consumir fácilmente toda la ventana de salida, lo que resulta especialmente problemático cuando la generación se corta bruscamente debido al límite de tokens. Este truncamiento a menudo da como resultado un JSON no válido, al que le faltan corchetes o de cierre cruciales, lo que hace que la salida sea inutilizable. Afortunadamente, herramientas como la biblioteca `json-repair` (disponible en PyPI) pueden ser de gran ayuda en estas situaciones. Esta librería intenta arreglar automáticamente objetos JSON incompletos o malformados, convirtiéndola en un aliado crucial cuando se trabaja con JSON generado por LLM, especialmente cuando se trata de potenciales problemas de truncamiento.

## Trabajar con esquemas

Usar JSON estructurado como salida es una gran solución, como hemos visto varias veces en este artículo. ¿Pero qué pasa con *la entrada*? Aunque JSON es excelente para estructurar la *salida* que genera el LLM, también puede ser increíblemente útil para estructurar la *entrada* que proporcionas. Aquí es donde entran en los esquemas JSON. Un esquema JSON define la estructura esperada y los tipos de datos de tu entrada JSON. Al proporcionar un esquema, le das al LLM un esquema claro de los datos que debe esperar, ayudándole a centrar su *atención* en la información relevante y reduciendo el riesgo de malinterpretar la entrada. Además, los esquemas pueden ayudar a establecer relaciones entre diferentes piezas de datos e incluso hacer que el LLM sea "consciente del tiempo" al incluir campos de fecha y hora con formatos específicos.

He aquí un ejemplo sencillo:

Supongamos que desea utilizar un LLM para generar descripciones de productos en un catálogo de comercio electrónico. En lugar de limitarse a proporcionar una descripción de texto libre del producto, puede utilizar un esquema JSON para definir los atributos del producto:

```
{
  "tipo": "object",
  "properties": {
    "name": { "type": "cadena", "descripción": "Nombre del producto" },
    "category": { "type": "cadena", "descripción": "Categoría del producto" },
    "precio": { "type": "number", "format": "float", "description": "Producto
precio" },
    "features": {
      "type": "array",
      "items": { "type": "string" }, "description":
      "Principales características del producto"
    },
    "release_date": {"type": "string", "format": "date", "description": "Fecha
de lanzamiento del producto"}
  },
}
```

Recorte 5. Definición del esquema de salida estructurado

A continuación, puede proporcionar los datos reales del producto como un objeto JSON que se ajuste a este esquema:

```
{
  "nombre": "Auriculares
  inalámbricos", "categoría":
  "Electrónica", "precio": 99.99,
  "características": ["Cancelación de ruido", "Bluetooth 5.0", Autonomía de 20
  horas"],
  "release_date": "2023-10-27"
```

Fragmento 6. Salida estructurada del LLM

Al preprocesar sus datos y, en lugar de proporcionar documentos completos, proporcionar únicamente el esquema y los datos, proporciona al LLM una comprensión clara de los atributos del producto, incluida su fecha de lanzamiento, lo que hace mucho más probable que genere una descripción precisa y relevante. Este enfoque de entrada estructurada, que guía la atención del LLM hacia los campos relevantes, es especialmente valioso cuando se trabaja con grandes volúmenes de datos o cuando se integran LLM en aplicaciones complejas.

## Experimenta con otros ingenieros

Si te encuentras en una situación en la que tienes que intentar crear un buen aviso, puede que quieras que varias personas lo intenten. Cuando todo el mundo sigue las mejores prácticas (como se indica en este capítulo) se va a ver una variación en el rendimiento entre todos los diferentes intentos de aviso.

## CoT Buenas prácticas

En el caso de CoT, es necesario poner la respuesta después del razonamiento porque la generación del razonamiento cambia los tokens que recibe el modelo cuando predice la respuesta final.

Con el CoT y la autoconsistencia tienes que ser capaz de extraer la respuesta final de tu pregunta, separada del razonamiento.

Para la solicitud de CoT, ajuste la temperatura a 0.

El razonamiento en cadena se basa en la descodificación codiciosa, que predice la siguiente palabra de una secuencia basándose en la mayor probabilidad asignada por el modelo lingüístico. En general, cuando se utiliza el razonamiento, para llegar a la respuesta final, es probable que haya una única respuesta correcta. Por lo tanto, la temperatura debe ser siempre 0.

## Documentar los distintos afeptos de aviso

El último consejo ya se ha mencionado antes en este capítulo, pero nunca se insistirá lo suficiente en lo importante que es: documenta tus intentos con todo detalle para que puedas aprender con el tiempo lo que ha ido bien y lo que no.

Los resultados de las preguntas pueden diferir entre modelos, entre configuraciones de muestreo e incluso entre distintas versiones del mismo modelo. Además, incluso entre instrucciones idénticas para el mismo modelo, pueden producirse pequeñas diferencias en el formato de la frase de salida y en la elección de palabras. (Por ejemplo, como se ha mencionado anteriormente, si dos fichas tienen la misma probabilidad de predicción, los empates pueden romperse al azar. Esto puede repercutir en los siguientes tokens predichos).

Recomendamos crear una hoja de Google con la Tabla 21 como plantilla. Las ventajas de este enfoque son que dispones de un registro completo cuando inevitablemente tengas que volver a revisar tu trabajo de orientación, ya sea para retomarlo en el futuro (te sorprendería lo mucho que puedes olvidar después de una pausa shofi), para probar el rendimiento rápido en diferentes versiones de un modelo, y para ayudar a depurar futuros errores.

Además de los campos de esta tabla, también es útil hacer un seguimiento de la versión de la solicitud (iteración), un campo para capturar si el resultado fue OK / NO OK / A VECES OK, y un campo para capturar la retroalimentación. Si tiene la suerte de utilizar Vefiex AI Studio, guarde sus avisos (utilizando la función mismo nombre y versión que figuran en su documentación) y siga el hipervínculo a la consulta guardada en la tabla. De este modo, siempre estará a un clic de volver a ejecutar sus avisos.

Cuando se trabaja en un sistema de *generación aumentada* de recuperación, también se deben capturar los aspectos específicos del sistema RAG que influyen en el contenido que se inyectó en la consulta, incluida la consulta, la configuración de los trozos, la salida de los trozos y otra información.

Una vez que sientas que el aviso está casi perfecto, llévalo a la base de código de tu proyecto. Y en la base de código, guarda los avisos en un archivo separado del código, para que sea más fácil de mantener. Por último, lo ideal es que los avisos sean parte de un sistema operacionalizado y, como ingeniero de avisos, debes basarte en pruebas automatizadas y procedimientos de evaluación para comprender qué tan bien se generaliza tu aviso a una tarea.

La ingeniería de instrucciones es un proceso iterativo. Elabore y pruebe distintas instrucciones, analícelas y documente los resultados. Perfeccione sus instrucciones basándose en el rendimiento del modelo. Siga experimentando hasta obtener el resultado deseado. Cuando cambie un modelo o la configuración de un modelo, vuelva atrás y siga experimentando con las instrucciones utilizadas anteriormente.

Nombre	[nombre y versión de su sistema]		
Objetivo	[Explicación en una frase del objetivo de este intento].		
Modelo	[nombre y versión del modelo utilizado]		
Temperatura	[valor entre 0 - 1]	Límite de fichas	[número]
Top-K	[número]	Top-P	[número]
Pregunte a	[Escriba todo el mensaje completo]		
Salida	[Escribe la salida o varias salidas]		

Tabla 21. Plantilla para documentar los avisos

## Resumen

Este libro blanco trata de la ingeniería de avisos. Aprendimos varias técnicas de incitación, tales como:

- Cero indicaciones
- Pocos disparos
- Indicaciones del sistema
- Indicación de funciones
- Indicaciones contextuales
- Preguntar con la mano
- Cadena de pensamiento
- Auto coherencia
- Árbol de pensamientos

- ReAct

Incluso hemos estudiado formas de automatizar los avisos.

A continuación, el informe analiza los retos de la IA genérica, como los problemas que pueden surgir cuando los avisos son insuficientes. Concluimos con las mejores prácticas para convertirse en un mejor ingeniero de avisos.

## Notas finales

1. Google, 2023, Gemini by Google. Disponible en: <https://gemini.google.com>.
2. Google, 2024, Gemini for Google Workspace Prompt Guide. Disponible en: <https://inthecloud.withgoogle.com/gemini-para-google-workspace-prompt-guide/dl-cd.html>.
3. Google Cloud, 2023, Introducción a los avisos. Disponible en: <https://cloud.google.com/vefiex-ai/generative-ai/docs/learn/prompts/introduction-prompt-design>.
4. Google Cloud, 2023, Text Model Request Body: Top-P & top-K sampling methods. Disponible en: [https://cloud.google.com/vefiex-ai/docs/generative-ai/model-reference/text#request\\_body](https://cloud.google.com/vefiex-ai/docs/generative-ai/model-reference/text#request_body).
5. Wei, J., et al., 2023, Zero Shot - Fine Tuned language models are zero shot learners. Disponible en: <https://arxiv.org/pdf/2109.01652.pdf>.
6. Google Cloud, 2023, Google Cloud Model Garden. Disponible en: <https://cloud.google.com/model-garden>.
7. Brown, T., et al., 2023, Few Shot - Los modelos lingüísticos son aprendices de Few Shot. Disponible en: <https://arxiv.org/pdf/2005.14165.pdf>.
8. Zheng, L., y otros, 2023, Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models. Disponible en: <https://openreview.net/pdf?id=3bq3jsvcQ1>
9. Wei, J., y otros, 2023, Chain of Thought Prompting. Disponible en: <https://arxiv.org/pdf/2201.11903.pdf>.
10. Google Cloud Platform, 2023, Cadena de pensamiento y React. Disponible en: [https://github.com/GoogleCloudPlatform/generative-ai/blob/main/language/prompts/examples/chain\\_of\\_thought\\_react.ipynb](https://github.com/GoogleCloudPlatform/generative-ai/blob/main/language/prompts/examples/chain_of_thought_react.ipynb).
11. Wang, X., et al., 2023, Self Consistency Improves Chain of Thought reasoning in language models. Disponible en: <https://arxiv.org/pdf/2203.11171.pdf>.
12. Yao, S., y otros, 2023, Tree of Thoughts: Deliberate Problem Solving with Large Language Models. Disponible en: <https://arxiv.org/pdf/2305.10601.pdf>.
13. Yao, S., y otros, 2023, ReAct: Synergizing Reasoning and Acting in Language Models. Disponible en: <https://arxiv.org/pdf/2210.03629.pdf>.
14. Google Cloud Platform, 2023, Prompting de Avance: Chain of Thought y React. Disponible en: [https://github.com/GoogleCloudPlatform/applied-ai-engineering-samples/blob/main/genai-on-vefiex-ai/advanced\\_prompting\\_training/cot\\_react.ipynb](https://github.com/GoogleCloudPlatform/applied-ai-engineering-samples/blob/main/genai-on-vefiex-ai/advanced_prompting_training/cot_react.ipynb).
15. Zhou, C., et al., 2023, Automatic Prompt Engineering - Large Language Models are Human-Level Prompt Engineers. Disponible en: <https://arxiv.org/pdf/2211.01910.pdf>.