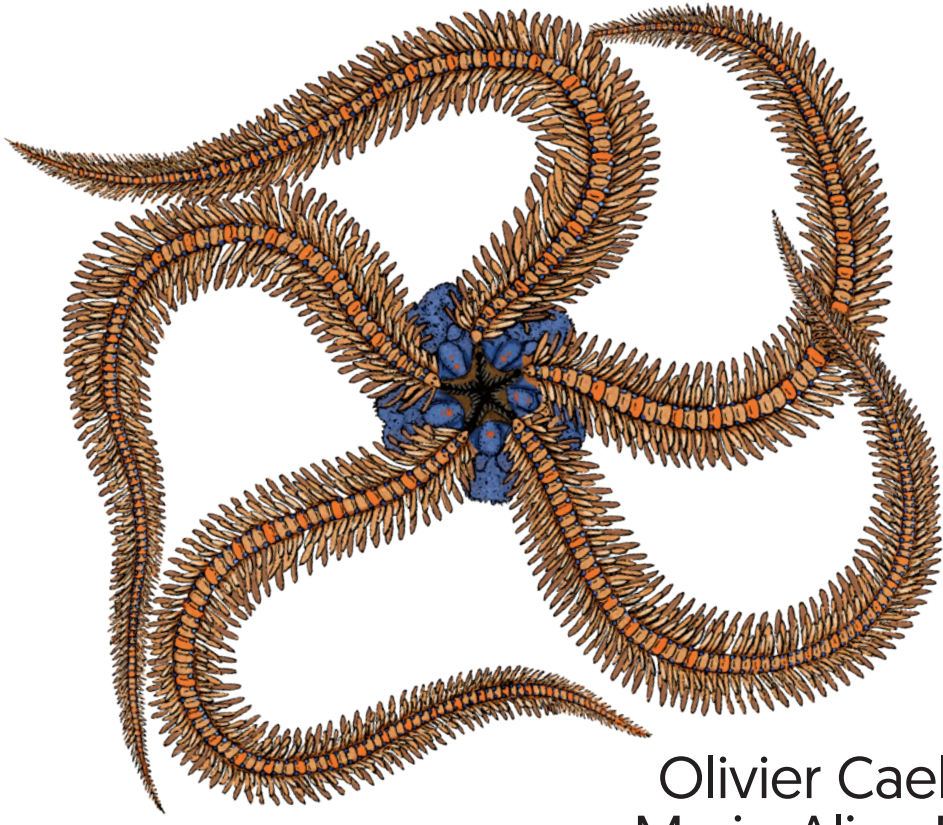
O'REILLY® ANAYA MULTIMEDIA

Desarrollo de aplicaciones con GPT-4 y ChatGPT

Cree *chatbots* inteligentes, generadores de contenido y mucho más



Olivier Caelen y
Marie-Alice Blete

Índice de contenidos

1 Conceptos básicos de GPT-4 y ChatGPT

Imagine un mundo donde puede comunicarse con los ordenadores tan rápido como con sus amigos. ¿Qué aspecto tendría? ¿Qué aplicaciones podría crear? Este es el mundo que OpenAI está ayudando a construir con sus modelos GPT, dando capacidades conversacionales similares a las de los humanos a nuestros dispositivos. Como avances más recientes de la IA, GPT-4 y otros modelos GPT son modelos grandes de lenguaje (*large language models*, LLM) entrenados en cantidades enormes de datos, lo que les permite reconocer y generar texto similar al de los humanos con una precisión muy alta.

Las implicaciones de estos modelos de IA van más allá de los simples asistentes de voz. Gracias a los modelos de OpenAI, ahora los desarrolladores pueden aprovechar el poder del procesamiento del lenguaje natural (PLN) para crear aplicaciones que entiendan nuestras necesidades de formas que en el pasado eran ciencia ficción. Desde innovadores sistemas de atención al cliente que aprenden y se adaptan hasta herramientas educativas personalizadas que entienden el estilo de aprendizaje único de cada estudiante, GPT-4 y ChatGPT abren todo un mundo nuevo de posibilidades.

Pero ¿qué son GPT-4 y ChatGPT? El objetivo de este capítulo es sumergirse en los cimientos, orígenes y características clave de estos modelos de IA. Al entender los conceptos básicos de estos modelos, podrá trabajar en la creación de la siguiente generación de aplicaciones que funcionen con IA.

Introducción a los modelos grandes de lenguaje

Esta sección expone los bloques de construcción fundamentales que han dado forma al desarrollo de GPT-4 y ChatGPT. Nuestro objetivo es proporcionar un entendimiento exhaustivo de los modelos de lenguaje y el PLN, el papel de las arquitecturas de transformadores y los procesos de tokenización y predicción dentro de los modelos GPT.

Explorar las bases de los modelos de lenguaje y el PLN

Como LLM, GPT-4 y ChatGPT son el último tipo de modelo obtenido en el campo del PLN, que es en sí mismo un subcampo del *machine learning* (ML) y la IA. Antes de profundizar en GPT-4 y ChatGPT, es esencial echar un vistazo al PLN y sus campos relacionados.

Hay diferentes definiciones de IA, pero una de ellas, más o menos el consenso, dice que la IA es el desarrollo de sistemas informáticos que pueden realizar tareas que suelen requerir inteligencia humana. Con esta definición, muchos algoritmos quedan dentro del concepto general de IA. Piense, por ejemplo, en la tarea de predicción del tráfico en aplicaciones de GPS o sistemas basados en reglas utilizados en videojuegos de estrategia. En estos ejemplos, vistos desde fuera, la máquina parece requerir inteligencia para completar estas tareas.

El ML es un subconjunto de la IA. En el ML, no intentamos implementar de forma directa las reglas de decisión utilizadas por el sistema de IA, sino que intentamos desarrollar algoritmos que permitan al sistema aprender por sí mismo a partir de ejemplos. Desde los años cincuenta, cuando comenzó la investigación del ML, se han propuesto muchos algoritmos de ML en la literatura científica.

Entre ellos, han pasado al primer plano los algoritmos de *deep learning*. El *deep learning* es una rama del ML que se centra en algoritmos inspirados en la estructura del cerebro. Esos algoritmos se denominan redes neuronales artificiales. Pueden manejar cantidades de datos muy grandes y desempeñar muy bien tareas como el reconocimiento de imágenes y de discurso y el PLN.

GPT-4 y ChatGPT se basan en un tipo particular de algoritmo de aprendizaje denominado transformador. Los transformadores son como máquinas de lectura. Prestan atención a diferentes partes de una frase o bloque de texto para entender su contexto y producir una respuesta coherente. También pueden entender el orden de las palabras en una oración y su contexto. Esto hace que sean muy efectivos en tareas como la traducción de idiomas, la respuesta a preguntas y la generación de texto. La figura 1.1 ilustra las relaciones entre estos términos.

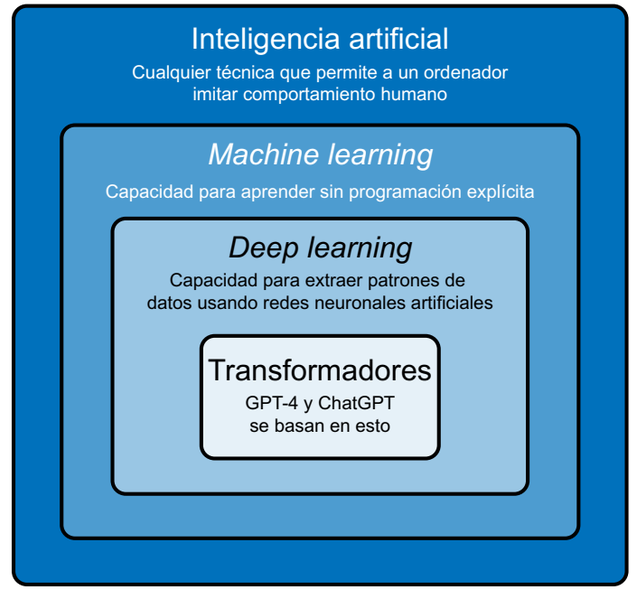


Figura 1.1. Un conjunto anidado de tecnologías desde la IA a los transformadores.

El PLN es un subcampo de la IA centrado en permitir a los ordenadores procesar, interpretar y generar lenguaje humano natural. Las soluciones de PLN modernas se basan en algoritmos de ML. El objetivo del PLN es permitir a los ordenadores procesar texto en lenguaje natural. Este objetivo abarca una amplia variedad de tareas:

- **Clasificación de textos:** Categorización del texto de entrada en grupos predefinidos. Esto incluye, por ejemplo, el análisis de sentimiento y la categorización de temas. Las empresas pueden usar el análisis de sentimiento para entender las opiniones de sus clientes acerca de sus servicios. La filtración de correos electrónicos es un ejemplo de categorización de temas, en la que un correo puede ponerse en categorías como "Personal", "Social", "Promoción" y "No deseado".
- **Traducción automática:** Traducción automática de texto de un lenguaje a otro. Tenga en cuenta que esto puede incluir áreas como traducir código de un lenguaje de programación a otro, como Python a C++.
- **Responder a preguntas:** Respuestas a preguntas basadas en un texto dado. Por ejemplo, un portal de atención al cliente en línea podría utilizar un modelo de PLN para responder preguntas frecuentes sobre un producto o un software

A continuación, definimos las especificaciones de las funciones:

```
# Definición de la función
functions = [
    {
        "name": "find_product",
        "description": "Get a list of products from a sql query",
        "parameters": {
            "type": "object",
            "properties": {
                "sql_query": {
                    "type": "string",
                    "description": "A SQL query",
                }
            },
            "required": ["sql_query"],
        },
    }
]
```

Entonces, podemos crear una conversación y llamar al *endpoint* `openai.ChatCompletion`:

```
# Pregunta de ejemplo
user_question = "I need the top 2 products where the price is less than 2.00"
messages = [{"role": "user", "content": user_question}]
# Llame al endpoint openai.ChatCompletion con la definición de la función
response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo-0613", messages=messages, functions=functions
)
response_message = response["choices"][0]["message"]
messages.append(response_message)
```

El modelo ha creado una consulta que podemos utilizar. Si imprimimos el objeto `function_call` de la repuesta, obtenemos:

```
"function_call": {
        "name": "find_product",
        "arguments": '{\n  "sql_query": "SELECT * FROM products \
    WHERE price < 2.00 ORDER BY price ASC LIMIT 2"\n}',
    }
```

A continuación, ejecutamos la función y continuamos la conversación con el resultado:

```
# Llame a la función
function_args = json.loads(
    response_message["function_call"]["arguments"]
)
products = find_product(function_args.get("sql_query"))
# Agregue la respuesta de la función a los mensajes
messages.append(
    {
        "role": "function",
        "name": function_name,
        "content": json.dumps(products),
    }
)
# Dé a la respuesta de la función formato de lenguaje natural
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo-0613",
    messages=messages,
)
```


Y, por último, extraemos la respuesta final y obtenemos lo siguiente:

```
The top 2 products where the price is less than $2.00 are:
1. Pen (Blue) - Price: $1.99
2. Pen (Red) - Price: $1.78
```

Este ejemplo sencillo muestra lo útiles que pueden ser las funciones para crear una solución que permita a los usuarios finales interactuar en lenguaje natural con una base de datos. Las definiciones de función le permiten restringir el modelo para que responda exactamente lo que usted quiera e integra su respuesta en una aplicación.

Utilizar otros modelos de compleción de texto

Como ya hemos mencionado, OpenAI proporciona varios modelos adicionales, además de las series GPT-3 y GPT-3.5. Estos modelos usan un *endpoint* diferente a los ChatGPT y GPT-4. En el momento de escribir esto, las versiones instantáneas más recientes eran: `gpt-3.5-turbo-0613`, `gpt-3.5-turbo-16k-0613`, `gpt-4-0613` y `gpt-4-32k-0613`.

> OpenAI ha marcado este *endpoint* como heredado.

Hay una diferencia importante entre la compleción de texto y la compleción de chat: como podrá suponer, ambas generan texto, pero la compleción de chat está optimizada para conversaciones. Como puede ver en el siguiente fragmento de código, la principal diferencia con el *endpoint* `openai.ChatCompletion` es el formato de los *prompts*. Los modelos basados en chat deben tener formato de conversación; para la compleción, es un solo *prompt*:

```
import openai
# Llame al endpoint openai Completion
response = openai.Completion.create(
    model="text-davinci-003", prompt="Hello World!"
)
# Extraiga la respuesta
print(response["choices"][0]["text"])
```

Proyecto 2: Resumir vídeos de YouTube

Los LLM han demostrado ser buenos a la hora de resumir textos. En la mayoría de los casos, consiguen extraer las ideas principales y reformular la entrada original de manera que el resumen generado resulte fluido y claro. La creación de resúmenes de textos puede ser útil en muchos casos:

- **Monitorización de medios:** Obtener una visión general rápida sin sobrecarga de información.
- **Observación de tendencias:** Generar *abstracts* de noticias de tecnología o agrupar artículos académicos y obtener resúmenes útiles.
- **Atención al cliente:** Generar visiones generales de documentación para que los clientes no se vean abrumados con información genérica.
- **Filtrar correo electrónico:** Hacer que aparezca la información más importante y evitar la sobrecarga de correos electrónicos.

Para este ejemplo, vamos a resumir vídeos de YouTube. Puede que se sorprenda: ¿cómo podemos introducir vídeos en modelos ChatGPT o GPT-4?

Bueno, aquí el truco está en considerar esta tarea como dos pasos distintos:

1. Extraer la transcripción del vídeo.
2. Resumir la transcripción del paso 1.

Puede acceder a la transcripción de un vídeo de YouTube de manera muy sencilla. Debajo del vídeo que elija ver, encontrará las acciones disponibles, como muestra la figura 3.2. Haga clic en la opción ... y, a continuación, elija Mostrar transcripción.

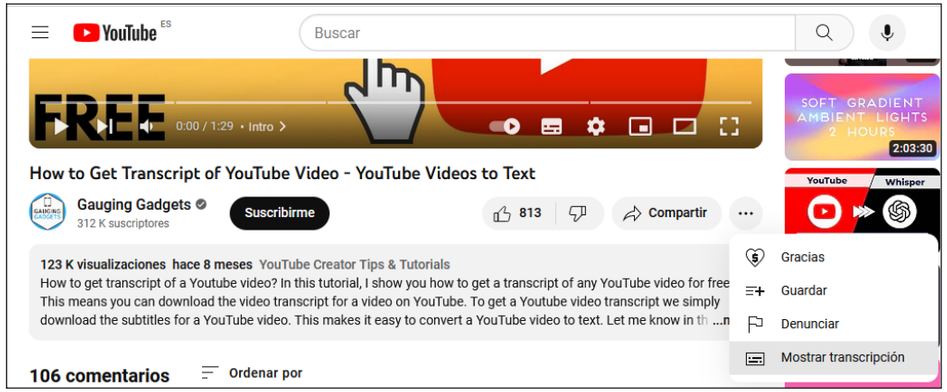


Figura 3.2. Acceso a la transcripción de un vídeo de YouTube.

Aparecerá un cuadro de texto que contiene la transcripción del vídeo; debería parecerse a la figura 3.3. Este cuadro también le permite activar y desactivar las marcas de tiempo.

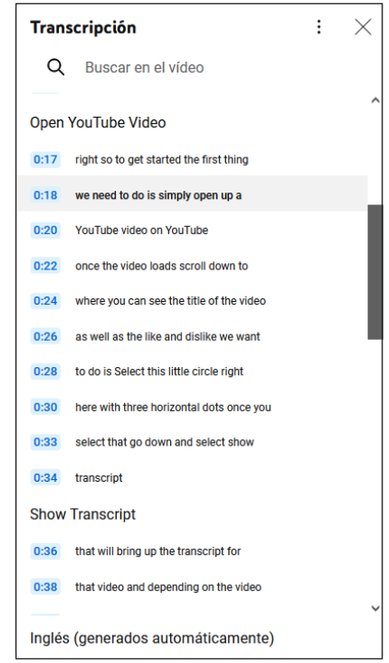


Figura 3.3. Ejemplo de transcripción de un vídeo de YouTube que explica las transcripciones de YouTube.

Si únicamente planea hacer esto una vez para un solo vídeo, podría simplemente copiar y pegar la transcripción que aparece en la página de YouTube. De lo contrario, necesitará utilizar una solución más automatizada, como la API (`https://oreil.ly/r-5qw`) proporcionada por YouTube que permite interactuar mediante programación con los vídeos. Puede utilizar esta API directamente, con los recursos `captions` (`https://oreil.ly/DNV3_`), o utilizar una biblioteca de terceros como youtube-transcript-api (`https://oreil.ly/rrXGW`) o una utilidad web como Captions Grabber (`https://oreil.ly/IZzad`).

Una vez que tenga la transcripción, necesita llamar a un modelo de OpenAI para hacer el resumen. Para esta tarea, utilizamos GPT-3.5 Turbo. Este modelo funciona muy bien para esta tarea simple y es el menos caro en el momento de escribir esto.

El siguiente fragmento de código pide al modelo que genere un resumen de una transcripción:

4 Técnicas avanzadas para GPT-4 y ChatGPT

Ahora que se ha familiarizado con los conceptos básicos de los LLM y la API de OpenAI, es hora de llevar sus habilidades al siguiente nivel. Este capítulo recoge estrategias potentes que le permitirán aprovechar el verdadero potencial de ChatGPT y GPT-4. Desde la ingeniería de *prompts* y los aprendizajes *zero-shot* y *few-shot* al ajuste de modelos para tareas específicas, este capítulo le proporcionará todo el conocimiento necesario para crear cualquier aplicación que pueda imaginar.

Ingeniería de *prompts*

Antes de sumergirnos en la ingeniería de *prompts*, vamos a repasar brevemente la función de compleción del modelo de chat, ya que en esta sección se utilizará de forma extensiva. Para hacer el código más compacto, definimos la función de la siguiente manera:

```
def chat_completion(prompt, model="gpt-4", temperature=0):
    res = openai.ChatCompletion.create(
        model=model,
        messages=[{"role": "user", "content": prompt}],
        temperature=temperature,
    )
    print(res["choices"][0]["message"]["content"])
```

Esta función recibe un *prompt* y muestra el resultado de la compleción en el terminal. El modelo y la temperatura son dos características opcionales configuradas por defecto como GPT-4 y 0, respectivamente.

Para hacer una demostración de la ingeniería de *prompts*, vamos a volver al texto de ejemplo "As Descartes said, I think therefore" ("Como dijo Descartes, pienso, luego existo"). Si esta entrada se pasa a GPT-4, es natural para el modelo completar la oración añadiendo de forma iterativa los *tokens* más probables:

```
chat_completion("As Descartes said, I think therefore")
```

Como resultado, obtenemos el siguiente mensaje de salida:

```
I am. This famous philosophical statement, also known as "Cogito, ergo sum,"
emphasizes the existence of the self through the act of thinking or doubting.
Descartes used this statement as a foundational principle in his philosophy,
arguing that one's own existence is the most certain and indubitable fact
that can be known.
```

La ingeniería de *prompts* es una disciplina emergente centrada en el desarrollo de las prácticas más recomendables para crear entradas óptimas para los LLM con el fin de producir salidas deseables de la forma más programática posible. Como ingeniero de IA, debe saber cómo interactuar con la IA para obtener resultados utilizables para sus aplicaciones, cómo hacer las preguntas correctas y cómo escribir *prompts* de calidad; trataremos todos estos temas en esta sección.

Debería tenerse en cuenta que la ingeniería de *prompts* puede afectar a los costes de utilizar la API de OpenAI. La cantidad de dinero que pagará por usar la API es proporcional al número de *tokens* que envíe a y reciba de OpenAI. Como hemos mencionado en el capítulo 2, se recomienda encarecidamente el uso del parámetro `max_token` para evitar sorpresas desagradables en las facturas.

Tenga en cuenta también que debería considerar los diferentes parámetros que puede utilizar en los métodos `openai`, ya que puede obtener resultados muy diferentes con el mismo *prompt* si usa parámetros como `temperature`, `top_p` y `max_token`.

Diseñar *prompts* efectivos

Muchas tareas pueden llevarse a cabo mediante *prompts*, incluidas la elaboración de resúmenes, la clasificación de textos, el análisis del sentimiento y las respuestas a preguntas. En todas estas tareas, es común definir tres elementos en el *prompt*: un rol, un contexto y una tarea, como muestra la figura 4.1.

No siempre se necesitan los tres elementos, y el orden puede cambiarse, pero, si el *prompt* está bien construido y los elementos están bien definidos, debería obtener buenos resultados. Tenga en cuenta que, incluso cuando se usan estos tres elementos, puede que para tareas complejas necesite utilizar técnicas más avanzadas, como el aprendizaje *zero-shot*, el aprendizaje *few-shot* o el ajuste. Veremos estas técnicas avanzadas más adelante en este capítulo.

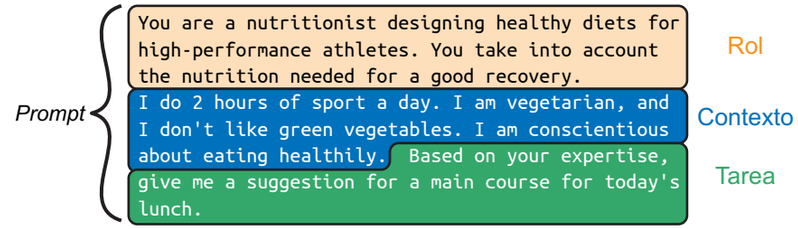


Figura 4.1. Un *prompt* efectivo.

El contexto

El primer elemento esencial de un *prompt* es el contexto. Cuando escribe un texto de entrada para un LLM, debe detallar el contexto tanto como sea posible. Para ilustrar esto, supongamos que quiere utilizar GPT-4 para crear una aplicación que sugiera el plato principal para una comida. A continuación, vamos a comparar los resultados que obtenemos con dos contextos diferentes. El primer contexto tendrá pocos detalles, y el segundo tendrá más.

Con un mensaje de entrada corto como:

```
chat_completion("Give me a suggestion for the main course for today's lunch.")
```

obtenemos el siguiente mensaje de salida:

```
Grilled chicken with roasted vegetables and quinoa.
```

Ahora, añadimos más detalles sobre el contexto en el mensaje de entrada (cuánto ejercicio hacemos, que somos vegetarianos, que no nos gustan las verduras de hoja):

```
prompt = """
Context: I do 2 hours of sport a day. I am vegetarian, and I don't like green
vegetables. I am conscientious about eating healthily.
Task: Give me a suggestion for a main course for today's lunch."""
chat_completion(prompt)
```

Y obtenemos el siguiente mensaje de salida, que sugiere un plato y da explicaciones al respecto:

```
For today's lunch, you can try a Chickpea and Sweet Potato Curry served with
Quinoa. This dish is packed with protein, fiber, and essential nutrients,
while also being vegetarian and not relying on green vegetables. The curry
can be made with chickpeas, sweet potatoes, tomatoes, onions, garlic, and a
blend of spices like turmeric, cumin, and coriander. Cook the quinoa
separately and serve it alongside the curry for a delicious and healthy meal.
```

En el segundo ejemplo, la oración es más detallada porque el modelo tiene más contexto: sugiere un plato vegetariano cargado de proteínas.

Desarrollo de aplicaciones con GPT-4 y ChatGPT

Este libro es una guía completa para desarrolladores de Python que quieren aprender a crear aplicaciones con grandes modelos de lenguaje. Los autores Olivier Caelen y Marie-Alice Blete abordan las características y beneficios principales de GPT-4 y ChatGPT y explican cómo funcionan. También es una guía paso a paso para desarrollar aplicaciones utilizando la biblioteca de Python para GPT-4 y ChatGPT, incluyendo herramientas de generación de texto, preguntas y respuestas y resumen de contenidos.

Escrito con un lenguaje claro y conciso, *Desarrollo de aplicaciones con GPT-4 y ChatGPT* incluye ejemplos fáciles de seguir para ayudarle a entender y aplicar los conceptos a sus proyectos. Los ejemplos de código de Python están disponibles en un repositorio de GitHub y el libro incluye un glosario de términos clave. ¿Listo para aprovechar el poder de los grandes modelos de lenguaje en sus aplicaciones? Este libro es de lectura obligatoria.

Aprenderá:

- Los fundamentos y beneficios de ChatGPT y GPT-4 y su funcionamiento.
- Cómo integrar estos modelos en aplicaciones basadas en Python para tareas de procesamiento del lenguaje natural.
- Cómo desarrollar aplicaciones usando las API de GPT-4 o ChatGPT en Python para generar textos, responder a preguntas y resumir contenidos, entre otras tareas.
- Temas de GPT avanzados, incluyendo la ingeniería de *prompts*, el ajuste de modelos para tareas específicas, *plugins*, LangChain y más.

«Mediante ejemplos prácticos e instrucciones paso a paso, los autores muestran el camino hacia el desarrollo de aplicaciones de vanguardia».

—Tom Taulli
Autor de *Generative AI* (Apress)

«Mezcla perfectamente la teoría y la práctica, haciendo que la complejidad de GPT-4 y ChatGPT resulte accesible».

—Lucas Soares
Ingeniero de ML, Biometrid

Olivier Caelen es investigador de ML en Worldline, empresa pionera en *paytech*. Tiene un doctorado en *machine learning* y también imparte cursos de ML en la Universidad de Bruselas.

Marie-Alice Blete es arquitecta de software e ingeniera de datos en el departamento de I+D en Worldline. También es *developer advocate* y oradora sobre tecnología.

ANAYA MULTIMEDIA O'REILLY®

www.anayamultimedia.es

ISBN 978-84-415-4925-8

2315256