Introducción a la Inteligencia Artificial

La inteligencia artificial (IA) es una disciplina tecnológica enfocada en dotar a las máquinas de la capacidad de realizar tareas que tradicionalmente requerirían inteligencia humana. Estas tareas abarcan desde el reconocimiento de voz y la toma de

decisiones hasta el procesamiento del lenguaje natural y el aprendizaje autónomo. En esencia, la IA busca imitar aspectos clave de la cognición humana, como la resolución de problemas y el aprendizaje, para mejorar la eficiencia y precisión en diversas actividades.

Actualmente, la IA se ha consolidado como una herramienta transversal en múltiples industrias, incluyendo la medicina, las finanzas y la educación. Su capacidad para predecir eventos, detectar anomalías y automatizar procesos complejos ha demostrado ser un catalizador para incrementar la productividad y optimizar flujos de trabajo.

Un aspecto destacado de la IA es su amplia aplicabilidad, que permite desde automatizar tareas repetitivas hasta asistir en procesos creativos. Por ejemplo, los modelos de inteligencia artificial generativa son capaces de producir contenido nuevo y original, como textos, imágenes y música, utilizando patrones extraídos de datos previamente analizados. Esta habilidad no solo facilita el desarrollo de soluciones innovadoras, sino que también amplía las posibilidades de personalización y adaptación de las herramientas tecnológicas a necesidades específicas.

El creciente uso de la inteligencia artificial, ejemplificado por herramientas como ChatGPT, ha democratizado el acceso a estas tecnologías avanzadas. Con interfaces intuitivas y versiones gratuitas que ofrecen funcionalidades significativas, estas herramientas han llegado a una amplia audiencia de profesionales y usuarios cotidianos. En consecuencia, la IA se está convirtiendo rápidamente en un recurso indispensable para el progreso en un mundo cada vez más digitalizado.

IA Generativa y Modelos de Lenguaje Extenso (LLMs)

La **IA Generativa** es un campo específico dentro de la inteligencia artificial que se enfoca en la creación de contenido original. A diferencia de los modelos tradicionales, que solo analizan datos existentes, los modelos generativos tienen la capacidad de producir textos, imágenes, música y otros tipos de contenido completamente nuevos. Estos modelos son entrenados con grandes volúmenes de datos y aprenden a generar salidas que imitan patrones presentes en los datos originales.

Por ejemplo, los modelos generativos pueden:

- Redactar artículos coherentes.
- Crear imágenes realistas a partir de descripciones.
- Componer música original.

Modelos de Lenguaje Extenso (LLMs)

En el ámbito de la IA generativa, los **Modelos de Lenguaje Extenso** (LLMs, por sus siglas en inglés) son una de las tecnologías más destacadas. Estos modelos, como los desarrollados por OpenAI, están diseñados para comprender, procesar y generar lenguaje natural de manera coherente.

Un **LLM** es un tipo de modelo de inteligencia artificial entrenado con grandes cantidades de texto. Su objetivo principal es manejar y generar lenguaje natural con un alto nivel de precisión. Ejemplos de estas capacidades incluyen:

- Responder preguntas.
- Resumir textos extensos.
- Realizar análisis de contenido de forma autónoma.

Características Principales de los LLMs

1. **Entrenamiento en datos masivos:** Los LLMs utilizan millones o incluso miles de millones de palabras en su entrenamiento, lo que les permite comprender una amplia variedad de temas y contextos.
2. **Capacidad de generar lenguaje natural:** Estos modelos producen respuestas coherentes y relevantes en función de las consultas que reciben.
3. **Versatilidad en tareas:** Pueden responder preguntas, redactar textos, traducir idiomas, generar código y más.

Casos de Uso de los LLMs

Los LLMs tienen aplicaciones en diversas industrias, como:

- **Medicina:** Ayudan a analizar datos médicos, generar informes y asistir en diagnósticos.
- **Finanzas:** Permiten predecir tendencias de mercado y detectar anomalías en tiempo real.
- **Educación:** Facilitan el aprendizaje personalizado y la generación de contenido educativo.
- **Automatización de tareas:** Integrados en herramientas como ChatGPT, estos modelos aumentan la productividad en tareas repetitivas o de alto volumen.

Limitaciones y Consideraciones

A pesar de sus beneficios, los LLMs tienen ciertas limitaciones:

- **Alucinaciones:** Pueden generar información incorrecta o irrelevante en algunos casos.

- **Dependencia del idioma:** Estos modelos suelen ser más precisos en inglés debido a la mayor cantidad de datos disponibles en ese idioma.
- **Costo computacional:** Manejar modelos de lenguaje extenso requiere una infraestructura significativa, especialmente en implementaciones personalizadas o a gran escala.

OpenAI y ChatGPT

Uno de los ejemplos más reconocidos de LLMs es **GPT** de OpenAI, la base de herramientas como ChatGPT. Este modelo ha pasado por varias iteraciones, cada una mejorando en términos de precisión, memoria de contexto y capacidad de generación. OpenAI también ofrece una API que permite integrar estos modelos en aplicaciones personalizadas, lo que abre un sinfín de posibilidades para desarrolladores y empresas.

Ingeniería de Prompts

La **ingeniería de prompts** es un aspecto fundamental en el uso de modelos de inteligencia artificial como ChatGPT y otros modelos de lenguaje extenso (LLMs). Los *prompts* son las instrucciones o consultas que los usuarios envían a estos modelos para obtener respuestas o realizar tareas específicas. La forma en que se diseña un prompt tiene un impacto directo en la calidad, precisión y relevancia de las respuestas generadas.

Concepto de Prompt

Un **prompt** es una instrucción escrita que guía al modelo de IA en su respuesta. Este puede ser una pregunta, un conjunto de instrucciones o incluso un contexto más amplio que ayude a establecer el tono y el objetivo de la interacción. Por ejemplo:

- Prompt simple: *"¿Qué es un prompt?"*
- Prompt detallado: *"Explica qué es un prompt usando ejemplos y un lenguaje técnico adecuado para un estudiante de secundaria."*

Importancia de los Prompts

El diseño de un buen prompt es clave para:

1. **Maximizar la relevancia:** Permite obtener respuestas precisas y alineadas con la intención del usuario.
2. **Optimizar recursos:** Un prompt bien estructurado reduce el consumo innecesario de tokens, haciendo que las interacciones sean más eficientes.

3. **Personalizar respuestas:** Con prompts adecuados, es posible adaptar las respuestas al nivel de conocimiento del usuario o a un estilo específico.

Elementos Claves en la Ingeniería de Prompts

1. **Claridad:** Un prompt debe ser claro y directo. Evitar la ambigüedad es crucial para obtener respuestas precisas.
    a. Ejemplo vago: *"Dime algo sobre tecnología."*
    b. Ejemplo claro: *"Explícame cómo funciona el aprendizaje supervisado en la inteligencia artificial."*
2. **Contexto:** Proporcionar un contexto relevante ayuda al modelo a entender mejor la consulta.
    a. Ejemplo sin contexto: *"Explica el proceso."*
    b. Ejemplo con contexto: *"Describe el proceso de entrenamiento de un modelo de lenguaje como ChatGPT."*
3. **Especificidad:** Ser específico sobre lo que se busca en la respuesta mejora la calidad de la información proporcionada.
    a. Ejemplo general: *"Genera una lista de tareas."*
    b. Ejemplo específico: *"Genera una lista de tareas para el mantenimiento de un hogar, enumeradas y organizadas por prioridad."*

Técnicas Avanzadas de Ingeniería de Prompts

1. **Asignación de roles:** Indicar al modelo que actúe con una personalidad o enfoque específico.
    a. Ejemplo: *"Eres un experto en inteligencia artificial. Explica el concepto de overfitting de manera técnica."*
2. **Delimitadores:** Utilizar delimitadores como comillas triples o corchetes angulares para estructurar el contenido o destacar partes relevantes.
    a. Ejemplo: *"""Describe el concepto de un token en el contexto de un modelo de IA."""*
3. **Control de longitud:** Definir el límite de palabras o tokens para la respuesta.
    a. Ejemplo: *"Explícame qué es un prompt en menos de 50 palabras."*
4. **Estructuración de la respuesta:** Solicitar un formato específico, como una lista, tabla o JSON.
    a. Ejemplo: *"Devuelve una lista de tareas en formato JSON con claves 'nombre' y 'prioridad'."*

Optimización Mediante Pruebas y Ajustes

La ingeniería de prompts implica iteración. A menudo, es necesario probar varios enfoques, evaluar las respuestas y ajustar el diseño del prompt hasta obtener el resultado deseado.

- **Prueba básica:** Comenzar con un prompt general.

- **Iteración:** Refinar el prompt según la calidad de las respuestas.
- **Evaluación:** Comparar diferentes formulaciones para determinar cuál es la más efectiva.

Conceptos Relacionados

1. **Tokens:** Los tokens son las unidades de texto procesadas por los modelos de IA. Diseñar prompts que minimicen el uso innecesario de tokens puede reducir costos y mejorar la eficiencia.
2. **Contexto:** Los modelos tienen un límite de memoria para recordar la conversación previa (definido en tokens). Es importante estructurar las interacciones de manera que el contexto más relevante se mantenga dentro de ese límite.

Errores Comunes en el Diseño de Prompts

- Ser demasiado general o ambiguo.
- Proporcionar un contexto insuficiente.
- No especificar el formato deseado de la respuesta.
- Hacer preguntas compuestas o confusas.

Root Prompts

Los **Root Prompts** son instrucciones iniciales o configuraciones base que se le proporcionan a un modelo de inteligencia artificial para definir su comportamiento general antes de interactuar con el usuario. Estas configuraciones son esenciales para personalizar y limitar las respuestas del modelo según las necesidades o restricciones de un contexto particular.

¿Qué es un Root Prompt?

Un **Root Prompt** es una serie de reglas o parámetros predefinidos que determinan cómo debe actuar el modelo de IA, qué tipo de información puede proporcionar y qué enfoque debe adoptar en sus respuestas. Este conjunto de instrucciones actúa como una "guía de comportamiento" para el modelo.

Propósitos de los Root Prompts

1. **Definir el alcance de las respuestas:**
    a. Restringen los temas o enfoques que el modelo puede abordar.
    b. Por ejemplo, limitarlo a responder exclusivamente preguntas técnicas o educativas.

2. **Proteger información sensible:**
    a. Evitan que el modelo comparta o utilice información confidencial.
3. **Ajustar el tono o nivel de formalidad:**
    a. Permiten establecer un estilo de respuesta acorde al público objetivo.
4. **Cumplir con normativas o políticas:**
    a. Aseguran que el modelo se alinee con regulaciones o restricciones específicas de una organización.

Ejemplo de Configuración de Root Prompts

- **Instrucción básica:**

"Responde a todas las consultas en un máximo de 100 palabras, utilizando un lenguaje formal y técnico."

- **Instrucción restringida:**

"No proporciones información relacionada con temas sensibles como seguridad cibernética o farmacología."

Control del Comportamiento del Modelo

El control del comportamiento a través de los Root Prompts es esencial para garantizar que las interacciones sean seguras, eficientes y alineadas con los objetivos del usuario. Algunas formas en que se puede implementar este control incluyen:

1. **Personalización de Respuestas:** Los modelos permiten modificar sus respuestas mediante configuraciones personalizadas. Esto puede lograrse accediendo a menús de configuración específicos, como en las herramientas avanzadas de ChatGPT, donde se definen estilos de respuesta predeterminados.
2. **Limitaciones Temáticas:** Los Root Prompts pueden restringir los temas que el modelo aborda, por ejemplo:
    a. Evitar responder preguntas relacionadas con temas no éticos.
    b. Redirigir consultas no deseadas hacia respuestas neutrales o informativas.
3. **Integración con Políticas de Seguridad:**
    a. Se pueden diseñar Root Prompts que limiten el almacenamiento de datos sensibles o que deshabiliten la memoria del modelo en sesiones específicas (como en los *chats temporales* de ChatGPT).

Ventajas del Uso de Root Prompts

- **Mayor precisión:** Al establecer un marco definido de actuación, las respuestas del modelo son más consistentes.
- **Personalización escalable:** Permite adaptar el comportamiento del modelo a diferentes contextos de uso, como educación, finanzas o atención al cliente.
- **Reducción de errores:** Ayudan a evitar respuestas incorrectas, inadecuadas o fuera del contexto deseado.

Limitaciones

1. **Rigidez en respuestas:**
    a. Si el Root Prompt es demasiado restrictivo, puede limitar la flexibilidad del modelo para adaptarse a consultas complejas.
2. **Configuración inicial:**
    a. Requiere un diseño detallado y una comprensión clara del propósito del modelo para establecer Root Prompts efectivos.
3. **Dependencia del diseño:**
    a. Un Root Prompt mal formulado puede generar respuestas inconsistentes o incompletas.

Seguridad y Privacidad en el Uso de IA

La seguridad y privacidad son aspectos fundamentales al utilizar modelos de inteligencia artificial, especialmente en contextos donde se manejan datos confidenciales o información crítica. Los modelos de IA procesan grandes cantidades de datos, y la forma en que estos datos son gestionados puede tener implicaciones importantes para los usuarios.

Riesgos Asociados al Uso de IA

1. **Exposición de Información Confidencial:**
    a. Las interacciones con modelos de IA, si no están adecuadamente protegidas, pueden ser utilizadas para entrenar futuros modelos. Esto implica un riesgo para datos sensibles compartidos durante las consultas.
2. **Fugas de Información:**
    a. Los modelos pueden recordar datos introducidos en sesiones anteriores si no se gestiona adecuadamente la memoria de contexto.
3. **Respuestas No Seguras:**
    a. Aunque el modelo intenta cumplir con las normativas éticas, existe la posibilidad de generar respuestas que expongan información

inapropiada o sensible debido a un diseño insuficiente en sus configuraciones.

Prácticas Recomendadas para Proteger la Seguridad y Privacidad

1. **Evitar Compartir Datos Sensibles:**
    a. No introducir información personal, confidencial o empresarial crítica durante las consultas.
    b. Por ejemplo, en lugar de incluir datos como "números de identificación fiscal" en una consulta, usar referencias genéricas.
2. **Usar Chats Temporales:**
    a. Muchas herramientas, como ChatGPT, ofrecen modos de chat temporal que no almacenan interacciones ni utilizan datos compartidos para entrenar futuros modelos. Esto es ideal para manejar información sensible.
3. **Deshabilitar la Memoria del Modelo:**
    a. Configurar el modelo para que no conserve un historial de interacciones puede prevenir el almacenamiento accidental de datos privados.
4. **Definir Instrucciones de Uso:**
    a. Utilizar configuraciones como Root Prompts para limitar el acceso del modelo a temas o datos específicos.
5. **Encriptación y Protocolos de Seguridad:**
    a. Garantizar que las conexiones utilizadas para interactuar con el modelo sean seguras, utilizando HTTPS y otros métodos de cifrado.

Manejo de Datos para Entrenamiento

Los datos proporcionados a modelos de IA pueden ser utilizados para mejorar su entrenamiento, salvo que el usuario limite este uso. Esto resalta la importancia de:

- Leer y comprender las políticas de privacidad del proveedor del modelo.
- Optar por configuraciones que limiten o excluyan el uso de datos para entrenamiento.

Cumplimiento de Normativas

1. **Regulaciones Locales e Internacionales:**
    a. Asegurar que el uso de IA cumpla con normativas de privacidad, como el GDPR en Europa o la CCPA en Estados Unidos, que regulan cómo se procesan y almacenan los datos personales.

2. **Auditorías de Seguridad:**
    a. Implementar revisiones regulares de los sistemas de IA para identificar posibles vulnerabilidades y áreas de mejora.

Modelos de IA Locales como Alternativa

En contextos donde la privacidad es crítica, una alternativa es el uso de **modelos de IA locales o de código abierto**, como los ofrecidos por plataformas como Hugging Face:

- Estos modelos pueden descargarse y ejecutarse en entornos controlados, eliminando la necesidad de enviar datos a servidores externos.
- Permiten un control total sobre los datos y las consultas, garantizando un nivel más alto de seguridad.

Hugging Face

Hugging Face es una plataforma líder en inteligencia artificial (IA) y aprendizaje automático (ML), ampliamente reconocida por su ecosistema de modelos de lenguaje y herramientas diseñadas para el desarrollo, implementación y uso de IA de forma accesible y eficiente. Es especialmente popular por facilitar el acceso a modelos de lenguaje extensos (LLMs) y otras arquitecturas avanzadas de IA de código abierto.

Características Principales de Hugging Face

1. **Model Hub:**
    a. Un repositorio centralizado que alberga miles de modelos de IA preentrenados.
    b. Los modelos abarcan una amplia gama de tareas como:
        i. Procesamiento del lenguaje natural (NLP): traducción, generación de texto, clasificación.
        ii. Visión por computadora (CV): detección de objetos, segmentación de imágenes.
        iii. Reconocimiento de voz y generación de audio.
    c. Los usuarios pueden buscar, descargar y usar modelos populares como **GPT-2**, **BERT**, **RoBERTa** y otros.
2. **Transformers:**
    a. Una biblioteca de código abierto para trabajar con arquitecturas de modelos basados en transformers.
    b. Compatible con lenguajes de programación como **Python** y marcos como **TensorFlow** y **PyTorch**.

    c. Permite implementar modelos de última generación con pocas líneas de código.
3. **Datasets:**
    a. Una biblioteca integrada con una gran colección de conjuntos de datos para entrenar y evaluar modelos.
    b. Incluye datos listos para tareas como clasificación, traducción, análisis de sentimiento, etc.
4. **Spaces:**
    a. Una plataforma para crear e implementar aplicaciones interactivas utilizando modelos de IA.
    b. Compatible con marcos como **Gradio** y **Streamlit** para crear interfaces gráficas rápidas y personalizables.
5. **Instrucciones de Personalización:**
    a. Ofrece herramientas para ajustar modelos preentrenados mediante técnicas como **fine-tuning**.
    b. Esto permite a las empresas y desarrolladores adaptar modelos a casos de uso específicos.

Ventajas de Hugging Face

1. **Accesibilidad:**
    a. Modelos y herramientas disponibles de forma gratuita, fomentando la democratización de la IA.
2. **Interoperabilidad:**
    a. Compatible con múltiples marcos y plataformas, lo que facilita la integración en proyectos existentes.
3. **Código Abierto:**
    a. La comunidad puede contribuir a mejorar las bibliotecas y compartir nuevos modelos y conjuntos de datos.
4. **Ecosistema Colaborativo:**
    a. Una comunidad activa que fomenta el aprendizaje y la innovación, con foros y documentación extensos.

Casos de Uso

1. **Procesamiento de Texto:**
    a. Clasificación de sentimientos en redes sociales.
    b. Resumen de documentos extensos.
    c. Traducción automática entre idiomas.
2. **Visión por Computadora:**
    a. Identificación de objetos en imágenes.

    b. Segmentación semántica en datos médicos.
3. **Desarrollo de Aplicaciones Interactivas:**
    a. Chatbots personalizados.
    b. Generación de contenido creativo, como arte o música.
4. **Investigación y Educación:**
    a. Pruebas rápidas de arquitecturas avanzadas para proyectos académicos y experimentación.

LMStudio.ai

LMStudio.ai es una plataforma y entorno diseñado para facilitar el acceso y la interacción con modelos de lenguaje avanzados en un entorno local. Es especialmente útil para quienes buscan utilizar inteligencia artificial sin depender de servicios en la nube, garantizando mayor privacidad, control y personalización. Es una solución atractiva para desarrolladores, empresas y entusiastas de la IA que valoran la seguridad de los datos y desean implementar modelos personalizados en sus propias máquinas.

Características Principales de LMStudio.ai

1. **Ejecución Local:**
    a. Permite descargar y ejecutar modelos de lenguaje directamente en tu computadora o servidor local.
    b. Ideal para entornos donde la privacidad y el control de datos son esenciales.
2. **Compatibilidad con Modelos Preentrenados:**
    a. Compatible con una amplia variedad de modelos disponibles en plataformas como Hugging Face.
    b. Soporta modelos grandes como GPT-2, LLaMA, BLOOM y otros.
3. **Interfaz Intuitiva:**
    a. Proporciona una interfaz gráfica fácil de usar para interactuar con modelos de lenguaje sin necesidad de programar.
    b. Los usuarios pueden realizar tareas como generación de texto, análisis de sentimientos y más, con un diseño amigable.
4. **Personalización y Fine-Tuning:**
    a. Ofrece herramientas para ajustar modelos preentrenados utilizando datasets personalizados.
    b. Esto permite adaptar los modelos a casos de uso específicos, como dominios técnicos o aplicaciones empresariales.

5. **Uso Offline:**
    a. Funciona completamente sin conexión a internet, lo que elimina la dependencia de servicios externos y garantiza que los datos procesados no se compartan con terceros.
6. **Soporte Multiplataforma:**
    a. Compatible con sistemas operativos populares como Windows, macOS y Linux, facilitando la adopción en diferentes entornos de trabajo.

Ventajas de LMStudio.ai

1. **Privacidad y Seguridad:**
    a. Al operar de manera local, los datos sensibles no se transmiten a servidores externos.
    b. Es ideal para industrias reguladas que manejan información confidencial, como salud, finanzas o derecho.
2. **Reducción de Costos:**
    a. Elimina la necesidad de pagar por servicios en la nube o suscripciones mensuales para el uso de modelos de IA.
    b. Una vez descargado el modelo, no hay costos adicionales por consultas.
3. **Personalización Total:**
    a. Ofrece flexibilidad para entrenar y ajustar modelos según necesidades específicas.
4. **Acceso Sin Restricciones:**
    a. No está limitado por las políticas o restricciones de uso impuestas por plataformas en la nube, como OpenAI o Google.

Casos de Uso

1. **Automatización Empresarial:**
    a. Automatización de tareas como redacción de correos, generación de reportes y asistencia en soporte técnico.
2. **Aplicaciones Educativas:**
    a. Creación de asistentes virtuales para tutorías personalizadas.
3. **Investigación:**
    a. Entrenamiento de modelos específicos para proyectos académicos o experimentos en IA.
4. **Desarrollo de Productos:**
    a. Integración de modelos en aplicaciones personalizadas sin depender de servicios externos.

Cómo Funciona LMStudio.ai

1. **Descarga del Entorno:**
    a. Los usuarios pueden descargar el software desde su sitio oficial y configurar el entorno localmente.
2. **Selección de un Modelo:**
    a. Se selecciona un modelo preentrenado desde una lista compatible o se sube uno propio.
3. **Interacción a través de la Interfaz:**
    a. Los usuarios pueden interactuar directamente con el modelo utilizando una interfaz gráfica intuitiva.
4. **Personalización (Opcional):**
    a. Si se requiere, se pueden ajustar los modelos con datasets específicos para optimizar los resultados.

Limitaciones de LMStudio.ai

1. **Requisitos Computacionales:**
    a. Algunos modelos avanzados requieren hardware potente, como GPUs con alto rendimiento, para un funcionamiento fluido.
2. **Tamaño de los Modelos:**
    a. Los modelos de lenguaje extensos pueden ocupar grandes cantidades de espacio en disco.
3. **Curva de Aprendizaje:**
    a. Aunque la interfaz es amigable, los ajustes avanzados como el fine-tuning pueden requerir conocimientos técnicos.

Comparación con Otras Plataformas

Aspecto	LMStudio.ai	Hugging Face	ChatGPT
Privacidad	Alta (uso local)	Media (depende del uso)	Baja (requiere nube)
Costo	Pago único o gratuito	Gratis para uso básico	Suscripción mensual

Requerimientos	Hardware local potente	Hardware y nube mixta	Sin requisitos locales
Personalización	Completa	Alta	Limitada

Consultas automatizadas a la API de Openai con Raspberry Pi

Introducción al uso de la API de ChatGPT

La API de ChatGPT es un recurso que permite automatizar interacciones con modelos de inteligencia artificial desarrollados por OpenAI. En este apartado, se detalla cómo iniciar el uso de esta herramienta, su configuración básica y su aplicación inicial mediante ejemplos prácticos.

1. Concepto y utilidad

Una API (Interfaz de Programación de Aplicaciones) es un mecanismo que facilita la comunicación entre dos programas informáticos. En este caso, permite conectar un programa desarrollado localmente (en lenguajes como Python) con los modelos de inteligencia artificial alojados en el entorno de OpenAI. Esto posibilita la realización de consultas automatizadas y la recepción de respuestas sin la necesidad de utilizar interfaces gráficas como ChatGPT directamente.

Por ejemplo, se puede usar la API para desarrollar un programa en una Raspberry Pi que automatice la interacción con ChatGPT, útil para tareas como la detección de correos maliciosos o la creación de asistentes personalizados.

2. Configuración inicial

El primer paso para usar la API de ChatGPT es crear una cuenta en la plataforma de OpenAI. Una vez registrada, es necesario realizar un depósito de crédito, ya que el uso de la API tiene un costo asociado basado en el número de tokens utilizados en consultas y respuestas.

Importante: OpenAI permite configurar límites de gasto diarios o mensuales para evitar costos inesperados, lo que brinda mayor control sobre el uso de la API.

3. Generación y uso de la API Key

La API Key es un código alfanumérico único que identifica al usuario y habilita la conexión con la API. Para obtenerla, se debe acceder al panel de control de OpenAI y

generar una clave. Es crucial copiar y guardar esta clave inmediatamente, ya que no podrá ser recuperada posteriormente.

En el código de ejemplo proporcionado, la clave API se almacena en una variable de Python para su reutilización:

```
import openai
# Reemplaza con tu clave API de OpenAI
openai.api_key = "Introduce tu APIKEY"
```

4. Aplicación inicial de la API

Una vez configurada la clave, es posible realizar consultas mediante solicitudes HTTP. En un ejemplo básico, se solicita un mensaje al usuario desde la terminal, se envía como consulta a la API y se imprime la respuesta:

```
def consulta_chatgpt(mensaje):
  try:
    respuesta = openai.ChatCompletion.create(
      model="gpt-3.5-turbo",
      messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": mensaje},
      ],
      max_tokens=150,
      temperature=0.7
    )
    return respuesta['choices'][0]['message']['content']
  except Exception as e:
    return f"Error en la consulta: {e}"
```

Este ejemplo establece parámetros clave como el modelo utilizado (gpt-3.5-turbo), el límite de tokens para la respuesta (max_tokens), y el nivel de creatividad de la respuesta (temperature).

Caso Práctico: Detección de Correos Maliciosos

En este apartado, exploraremos cómo utilizar la API de ChatGPT para detectar correos electrónicos maliciosos, centrándonos en el entrenamiento del modelo, la configuración del contexto y la ejecución de pruebas.

1. Entrenamiento y Configuración del Modelo

El entrenamiento del modelo consiste en definir un contexto y proporcionar ejemplos concretos que le permitan entender cómo identificar correos electrónicos maliciosos

(spam). Este proceso se inicia estableciendo un rol sistemático que describe al modelo como un especialista en ciberseguridad:

```
contexto_conversacion = [
  {"role": "system", "content": "You are an assistant trained to detect spam."},
  {"role": "user", "content": "Hola, quiero enseñarte cómo detectar correos electrónicos que son spam. Primero, vamos a revisar qué es spam."},
  {"role": "assistant", "content": "Claro, ¿podrías decirme qué características hacen que un correo electrónico sea considerado spam?"},
  {"role": "user", "content": "Un correo electrónico es spam si tiene muchos enlaces sospechosos, un tono urgente, y pide información personal o dinero. Los correos no solicitados de publicidad también suelen ser spam."},
  {"role": "assistant", "content": "Entendido, un correo electrónico es considerado spam si contiene enlaces sospechosos, un tono urgente, y solicita información personal o dinero. Además, los correos no solicitados de publicidad también pueden ser spam."}
]
```

Este contexto sirve como base para que el modelo aprenda a identificar características comunes en correos maliciosos, tales como:

- **Tono urgente:** Mensajes que intentan generar una reacción rápida.
- **Solicitudes sospechosas:** Peticiones de información personal, credenciales o dinero.
- **Enlaces peligrosos:** URLs que redirigen a sitios fraudulentos.

Además, se incluyen ejemplos prácticos para reforzar este aprendizaje.

2. Ejecución de Pruebas

Una vez configurado el contexto, se puede ejecutar una prueba. El correo que se desea analizar se añade al histórico de la conversación como una nueva entrada del usuario:

```
correo_a_analizar = input("Introduce el contenido del correo electrónico que quieres analizar para detección de spam: ")
contexto_conversacion.append({"role": "user", "content": f"¿Es spam el siguiente correo? {correo_a_analizar}"})
```

La consulta se envía a la API de ChatGPT, configurada para limitar los tokens de respuesta y asegurar una evaluación precisa:

```
respuesta = openai.ChatCompletion.create(
   model="gpt-3.5-turbo",
   messages=contexto_conversacion,
   max_tokens=50,  # Limita la longitud de la respuesta
   temperature=0.0  # Minimiza la aleatoriedad para una evaluación más precisa
)
respuesta_texto = respuesta['choices'][0]['message']['content']
```

El modelo devuelve una evaluación del correo, indicando si cumple con los criterios de un mensaje malicioso.

3. Análisis de Resultados

Al ejecutar una prueba con un correo como:

"Has ganado un iPhone, descarga el documento adjunto en menos de 10 minutos para no perder esta oportunidad."

El modelo responde con un análisis similar al siguiente:

"Ese correo también se considera spam. Promete un premio atractivo de forma poco realista, presiona al destinatario con un límite de tiempo y solicita la descarga de un documento adjunto."

Este resultado demuestra que el modelo puede identificar patrones de comportamiento típicos de correos maliciosos.

Hailo

El kit de Inteligencia Artificial incluye un adaptador llamado "M. Hat", que básicamente es un puente de dos carriles diseñado para conectar la placa del módulo M. Hilo con nuestra Raspberry Pi. Este módulo, conocido como M. Convcom, se coloca directamente sobre la Raspberry Pi mediante un zócalo, lo que permite establecer una conexión eficiente.

Gracias a esta configuración, podemos ejecutar en tiempo real inferencias de redes neuronales profundas, con un bajo consumo de energía, y aplicarlas a una amplia gama de segmentos de mercado. Entre las aplicaciones posibles están la detección, el seguimiento, el análisis de imágenes, el uso de modelos de lenguaje avanzados, y más.

El proceso es sencillo: basta con conectar el zócalo y montar el módulo en la placa. Además, en este proyecto utilizaremos una cámara compatible con la Raspberry Pi, que se conecta directamente a esta para llevar a cabo las tareas mencionadas.

Configuración Hailo

En esta clase vamos a aprender cómo configurar el kit de Inteligencia Artificial en una Raspberry Pi. Para comenzar, abrimos una terminal de comandos y actualizamos el sistema operativo utilizando los siguientes comandos:

```
sudo apt update
sudo apt upgrade
```

Estos comandos aseguran que el sistema operativo tenga las últimas actualizaciones instaladas.

El siguiente paso es activar el modo de velocidad avanzada que ofrece este dispositivo, optimizando así el rendimiento del kit de Inteligencia Artificial previamente instalado en nuestra placa Raspberry Pi.

Para ello, ejecutamos el comando:

```
sudo raspi-config
```

Esto abrirá una interfaz con un conjunto de opciones. Nos desplazamos hacia abajo hasta la opción número 6: **Advanced Options**. Una vez seleccionada, navegamos hasta la opción denominada **PCI Speed**. Dentro de esta sección, activamos la opción **Generation 3** para habilitar la velocidad de tercera generación.

Aceptamos los cambios y, para asegurarnos de que se apliquen correctamente, reiniciamos la Raspberry Pi ejecutando el siguiente comando:

```
sudo reboot
```

Con esto, habremos configurado correctamente el kit de Inteligencia Artificial para operar a máxima eficiencia en nuestra Raspberry Pi.

Redes neuronales

En esta sección aprenderemos cómo instalar, configurar y ejecutar ejemplos prácticos de redes neuronales utilizando el kit de Inteligencia Artificial HAILO IA junto con una Raspberry Pi.

1. Instalación del Repositorio de HAILO IA

El repositorio oficial de HAILO IA incluye una gran variedad de ejemplos listos para usar. Para instalarlo, seguimos estos pasos:

1. Abrimos una terminal y ejecutamos:

```
sudo apt install hailo
```

Seguimos las instrucciones en pantalla, aceptamos los términos y esperamos a que se complete la instalación. Este proceso puede tardar unos minutos, ya que se descargarán varios archivos necesarios.

2. Comprobamos que la instalación fue exitosa utilizando el siguiente comando:

```
hrt-cli fw-control identify
```

Este comando muestra la versión del software instalado junto con otros parámetros del sistema.

3. Verificamos la instalación del componente **GStreamer** necesario para usar el kit:

```
gst-inspect-1.0 hailo
```

Si la instalación fue exitosa, veremos un listado de herramientas disponibles.

2. Clonación del Repositorio de Ejemplos

Ahora descargaremos ejemplos prácticos directamente desde el repositorio oficial en GitHub:

1. Clonamos el repositorio utilizando:

```
git clone https://github.com/HailoAI/hailo_rpi_examples.git
```

2. Accedemos al directorio clonado:

```
cd hailo_rpi_examples
```

3. Ejecutamos el script de instalación de dependencias:

```
./install_dependencies.sh
```

Durante este proceso, se instalarán las librerías necesarias y recursos como modelos preentrenados, por ejemplo, **YOLO v3** para detección de objetos.

4. Activamos el entorno virtual para trabajar con los ejemplos:

```
source setup.sh
```

3. Ejecución de Ejemplos Prácticos

El repositorio incluye varios ejemplos básicos para probar redes neuronales en diferentes aplicaciones. A continuación, describimos algunos de ellos:

3.1Detección de Objetos

1. Para ejecutar el ejemplo de detección de objetos, usamos:

```
python3 basic_pipelines/detection.py
```

2. Este ejemplo utiliza la red YOLO para identificar objetos en tiempo real, como personas, bicicletas y vehículos. El programa procesa los fotogramas del video y muestra las detecciones tanto gráficamente como en la terminal.

3.2 Detección de Códigos de Barras

1. Para detectar códigos de barras, configuramos el programa para que use un modelo específico y un archivo JSON con las etiquetas:

```
python3 basic_pipelines/detection.py --labels resources/barcode-labels.json --video resources/barcode_sample.mp4
```

2. Este comando carga el video de ejemplo y utiliza un modelo entrenado para detectar exclusivamente códigos de barras.

3.3 Segmentación de Instancias

1. Para ejecutar la segmentación de instancias, usamos:

```
python3 basic_pipelines/instance_segmentation.py
```

2. Este ejemplo no solo detecta objetos, sino que los segmenta como instancias únicas, diferenciando entre elementos similares, como personas y vehículos.

3.4 Estimación de la Pose

Para estimar la pose de personas en un video, usamos:

```
python3 basic_pipelines/pose_estimation.py
```

El programa detecta y clasifica personas, dibujando un esqueleto sobre las imágenes para mostrar los puntos clave como cabeza, brazos y piernas.

4. Exploración del Código

Los ejemplos utilizan la librería **GStreamer**, ampliamente utilizada para el análisis de video. El flujo de trabajo incluye:

- Carga del video en un buffer de memoria.
- Procesamiento fotograma a fotograma mediante redes neuronales.
- Visualización de los resultados en pantalla y en la terminal.

4.1Detalles del Código de Detección

1. Importación del flujo de trabajo de GStreamer:

```
import gst_pipeline
```

2. Carga del video:

```
video_buffer = gst_pipeline.load_video('resources/sample.mp4')
```

3. Procesamiento fotograma por fotograma:

```
for frame in video_buffer:
   detections = gst_pipeline.detect(frame)
   gst_pipeline.render(detections, frame)
```

Aquí tienes el texto mejorado y reorganizado con explicaciones claras, comandos destacados y conceptos completados:

Uso de Ollama con el Kit de IA

En esta sesión exploraremos cómo utilizar Ollama, una herramienta que facilita la integración de modelos de lenguaje grande (LLM), junto con el kit de Inteligencia Artificial en una Raspberry Pi. También veremos cómo aprovechar sus capacidades para tareas avanzadas en IA.

1. Introducción a Ollama

Ollama es una herramienta diseñada para simplificar el uso de LLMs como GPT. Proporciona:

- **Facilidad de integración**: Añade capacidades de IA a aplicaciones y sitios web sin necesidad de desarrollar modelos desde cero.
- **Personalización**: Permite adaptar los modelos a necesidades específicas.
- **Escalabilidad**: Gestiona las complejidades técnicas de ejecutar LLMs.

2. Instalación de Ollama

2.1 Descargar y Configurar Ollama

1. Accedemos a la página oficial de Ollama.
2. Seleccionamos nuestro sistema operativo (en este caso, Linux/Ubuntu) y copiamos el enlace de descarga.
3. Abrimos una terminal y ejecutamos:

   ```
   wget <enlace_descarga>
   sudo dpkg -i <archivo_descargado>
   ```

4. Introducimos la contraseña cuando se solicite. Una vez completado, verificamos la instalación:ollama --version

2.2 Crear un Entorno Virtual en Python

Creamos y activamos un entorno virtual para gestionar las dependencias:

```
python3 -m venv ollama_env
source ollama_env/bin/activate
```

Ejecutar un Modelo

Descargamos y probamos un modelo de lenguaje como **Llama**:

```
ollama run llama --b
```

La opción --b especifica el tamaño del modelo, adaptado para Raspberry Pi.

1. Una vez descargado, interactuamos con el modelo haciendo preguntas. Por ejemplo:

> ¿Cuál es la capital de Francia?
Respuesta: París

Agregar el Modo Verbose

Para obtener métricas detalladas sobre las consultas:

ollama run llama --b --verbose

3. Ejemplos Prácticos

3.1Probar Diferentes Modelos

1. Ejecutamos modelos como **Gema** o **Fi** con:
   ```
   ollama run gema --b --verbose
   ```
2. Cada modelo tiene características únicas. Por ejemplo, Fi se especializa en privacidad al ejecutarse en local, evitando transferencia de datos a servidores externos.

3.2 Comparación de Modelos

Podemos evaluar el rendimiento de diferentes modelos:

1. Hacemos las mismas preguntas a múltiples modelos.
2. Analizamos las respuestas y métricas como tiempo de ejecución y precisión.

3.3Uso Multimodal

Ollama también soporta modelos que procesan texto e imágenes. Por ejemplo:

1. Utilizamos un modelo multimodal:

```
ollama run multimodal_model --image <ruta_imagen>
```

2. El modelo describe la imagen cargada, por ejemplo:
    a. **Imagen**: Una escena de París.
    b. **Descripción generada**: "Una vista del río Sena con la Torre Eiffel al fondo."

4. Uso de Ollama con Python

1. Instalamos las librerías necesarias:

```
pip install ollama matplotlib pillow
```

2. Ejecutamos un script en **Jupyter Notebook**:

```
jupyter-lab
```

En el entorno, importamos Ollama y cargamos modelos previamente descargados:

```
from ollama import Ollama

model = "gema"
response = Ollama.generate(model, prompt="¿Cuál es la capital de Francia?")
print(response['text'])
```

Trabajar en Modo Chat

Podemos mantener un contexto en las conversaciones:

```
chat_history = [
  {"role": "user", "content": "¿Cuál es la capital de Francia?"},
  {"role": "assistant", "content": "París"},
  {"role": "user", "content": "¿Cuál es la capital de España?"}
]

response = Ollama.chat(model, messages=chat_history)
print(response['text'])
```

Aquí tienes el texto mejorado y organizado, con explicaciones claras, comandos destacados y conceptos teórico-prácticos completados:

Uso de RAG (Generación Aumentada por Recuperación)

En esta sesión aprenderemos sobre RAG (siglas en inglés de **Retrieval-Augmented Generation**), un enfoque avanzado que combina la búsqueda de información relevante en bases de datos con la capacidad generativa de los modelos de lenguaje, como GPT.

1. ¿Qué es RAG?

RAG es un enfoque de Inteligencia Artificial que:

- **Recupera información**: Busca datos relevantes en bases de datos, documentos o recursos propios.
- **Genera texto**: Utiliza esa información para crear respuestas claras y coherentes.

Esto permite que, en lugar de depender solo de los datos aprendidos durante el entrenamiento, el modelo pueda consultar bases de datos externas y proporcionar respuestas actualizadas y específicas.

Componentes de RAG:

1. **Recuperador (Retriever)**: Encuentra información relevante en bases de datos o documentos. Funciona como un motor de búsqueda avanzado que comprende el significado detrás de las consultas.
2. **Generador (Generator)**: Redacta respuestas basándose en la información recuperada.
3. **Conexión (Bridge)**: Une ambos procesos para entregar una respuesta final adaptada al usuario.

1. ¿Por qué usar RAG en lugar de modelos tradicionales?

- **Actualización constante**: Accede a información en tiempo real, incluso después del entrenamiento del modelo.
- **Mayor precisión**: Reduce errores y "alucinaciones" (respuestas incorrectas o inventadas) al basarse en datos recuperados.
- **Personalización**: Permite usar bases de datos propias, adaptando el conocimiento a contextos específicos.

2. Ejemplo Práctico de RAG

Imaginemos un chatbot de soporte técnico:

- **Pregunta del usuario**: "¿Cómo restablezco mi contraseña?"

- **Proceso**:
    - El recuperador busca en una base de datos de preguntas frecuentes y encuentra una guía útil.
    - El generador redacta una respuesta como: "Para restablecer tu contraseña, ve a la página de inicio de sesión, haz clic en 'Olvidé mi contraseña', ingresa tu correo electrónico registrado y sigue las instrucciones."

Construyendo un RAG Local con Python

Paso 1: Instalación de Librerías

Primero instalamos las librerías necesarias:

```
pip install chromadb
pip install ollama
```

Paso 2: Preparación del Entorno

1. Creamos un entorno virtual para trabajar:

```
python3 -m venv rag_env
source rag_env/bin/activate
```

2. Descargamos el modelo para vectorizar las consultas:

```
ollama pull nomic-embed-text
```

Paso 3: Creación de una Base de Datos Vectorial

1. Utilizamos **ChromaDB** para guardar la información de manera vectorial:

```
import chromadb
from chromadb.config import Settings

client = chromadb.Client(Settings())
collection = client.create_collection("ejemplo")
```

2. Añadimos información a la base de datos:

```
data = [
  "La Raspberry Pi es una microcomputadora de bajo coste desarrollada en el Reino
```

```
Unido.",
   "Los pines GPIO permiten conectar sensores y dispositivos electrónicos.",
   "La Raspberry Pi puede conectarse a internet mediante WiFi o cable Ethernet."
]

for text in data:
   embedding = vectorizer.encode(text)  # Codificación en formato vectorial
   collection.add_document(embedding, metadata={"text": text})
```

Paso 4: Recuperación y Generación

1. Realizamos preguntas y buscamos respuestas en la base de datos:

```
query = "¿Cómo se conectan las Raspberry Pi a internet?"
results = collection.query(query, top_k=1)
```

2. Generamos una respuesta basada en los datos recuperados:

```
from ollama import Ollama

prompt = f"Usando la información: {results['metadata'][0]['text']}, responde a:
{query}"
response = Ollama.generate(model="gema", prompt=prompt)
print(response["text"])
```