«Talento Tech»

Inteligencia Artificial

Clase 07

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 7 | Introducción a Machine Learning y Groq

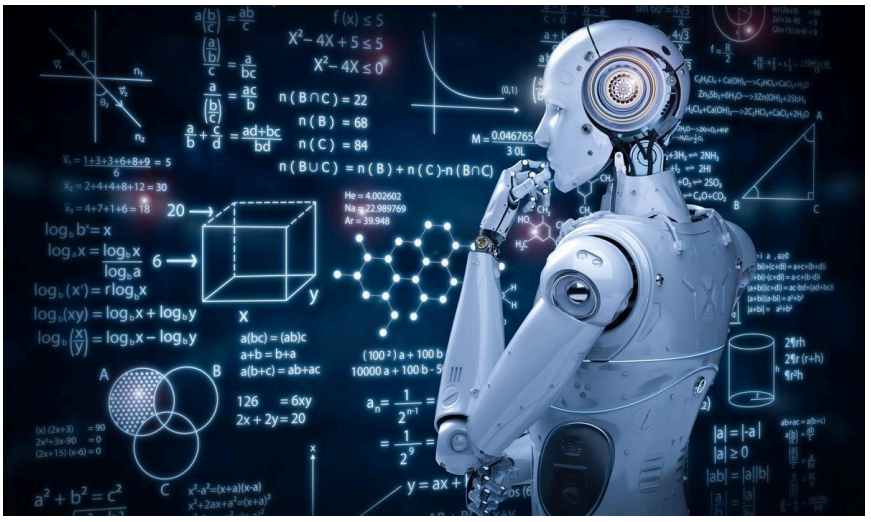
Temario:

- ¿Qué es Machine Learning? ¿Qué es Deep Learning? ¿Qué es un LLM?
- Aprendizaje supervisado vs no supervisado.
- Métricas a evaluar en un modelo de Inteligencia Artificial
- Configuración básica de un modelo de lenguaje (Groq).
- Introducción a las variables de estado en Streamlit.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

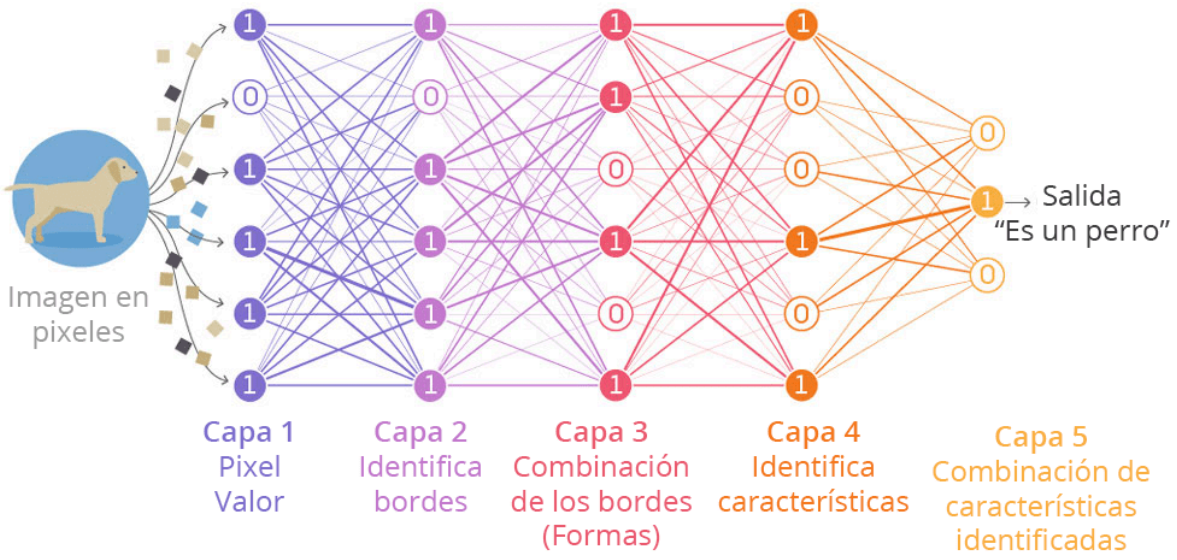
Machine Learning.

El **aprendizaje automático** es un subconjunto de la inteligencia artificial que permite que un sistema aprenda y mejore de forma autónoma con redes neuronales y aprendizaje profundo, sin necesidad de una programación explícita, a través del análisis de grandes cantidades de datos.
El **aprendizaje automático** permite que los sistemas informáticos se ajusten y mejoren continuamente a medida que acumulan más "experiencias". Por lo tanto, el rendimiento de estos sistemas puede mejorar si se proporcionan conjuntos de datos más grandes y variados para su procesamiento.



Deep Learning.

El **aprendizaje profundo** es un método de la inteligencia artificial (IA) que enseña a las computadoras a procesar datos de una manera que se inspira en el cerebro humano. Los modelos de aprendizaje profundo son capaces de reconocer patrones complejos en imágenes, textos, sonidos y otros datos, a fin de generar información y predicciones precisas. Es posible utilizar métodos de aprendizaje profundo para automatizar tareas que habitualmente requieren inteligencia humana, como la descripción de imágenes o la transcripción a texto de un archivo de sonido.

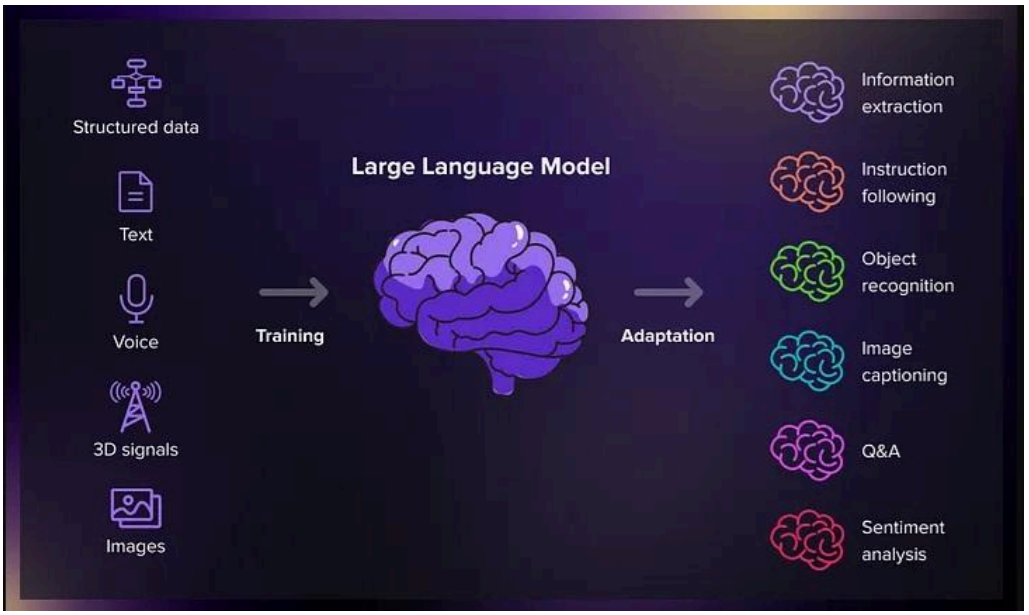


Fuente: https://www.quantamagazine.org/

Large Language Model (LLM)

Los **modelos de lenguaje de gran tamaño**, también conocidos como **LLM**, son modelos de aprendizaje profundo muy grandes que se preentrenan con grandes cantidades de datos. El transformador subyacente es un conjunto de redes neuronales que consta de un codificador y un decodificador con capacidades de autoatención. El codificador y el decodificador extraen significados de una secuencia de texto y comprenden las relaciones entre las palabras y las frases que contiene.

Los transformadores LLM son capaces de entrenarse sin supervisión, aunque una explicación más precisa es que los transformadores llevan a cabo un autoaprendizaje. Es a través de este proceso que los transformadores aprenden a entender la gramática, los idiomas y los conocimientos básicos.



¿Cuál es el futuro de los LLM?

La introducción de modelos de lenguaje de gran tamaño, como ChatGPT, Claude 2 y Llama 2, que pueden responder preguntas y generar texto, apunta a interesantes posibilidades en el futuro. De forma lenta pero segura, los LLM están logrando un rendimiento similar al humano. El éxito inmediato de estos LLM demuestra un gran interés en los LLM de tipo robótico que emulan y, en algunos contextos, superan al cerebro humano. A continuación, se mencionan algunas reflexiones sobre el futuro de los LLM:

Mayores capacidades: Por impresionantes que sean, el nivel tecnológico actual no es perfecto y los LLM no son infalibles. Sin embargo, las versiones más recientes mejorarán la precisión y las capacidades a medida que los desarrolladores aprendan a mejorar su rendimiento y, al mismo tiempo, reducir los sesgos y eliminar las respuestas incorrectas.

Entrenamiento audiovisual: Si bien los desarrolladores entrenan a la mayoría de los LLM con texto, algunos han empezado a entrenar modelos con entrada de video y audio. Este tipo de entrenamiento debería conducir a un desarrollo de modelos más rápido y abrir nuevas posibilidades en términos de uso de LLM para vehículos autónomos.

Transformación del lugar de trabajo: Los LLM son un factor disruptivo que cambiará el lugar de trabajo. Es probable que los LLM reduzcan las tareas monótonas y repetitivas de la misma manera que lo hicieron los robots con las tareas de fabricación repetitivas. Las

posibilidades incluyen tareas administrativas repetitivas, chatbots de servicio al cliente y redacción automatizada y simple de textos publicitarios.
IA conversacional: Sin duda, los LLM mejorarán el rendimiento de los asistentes virtuales automatizados como Alexa, Google Assistant y Siri. Podrán interpretar mejor la intención del usuario y responder a comandos sofisticados.

Aprendizaje Supervisado

El **aprendizaje supervisado** es una técnica de aprendizaje automático que se utiliza para entrenar sistemas informáticos con el objetivo de predecir resultados precisos.

Este tipo de aprendizaje se basa en un conjunto de datos de entrenamiento etiquetados, lo que significa que cada muestra de datos del conjunto de entrenamiento tiene una etiqueta que indica la respuesta correcta.

Para que te hagas una idea de su funcionamiento, el objetivo es que el modelo aprenda a encontrar patrones y relaciones en los datos sin tener información previa sobre ellos. En otras palabras, el modelo debe encontrar la estructura oculta en los datos sin ninguna guía.

El sistema informático analiza los datos de entrenamiento etiquetados y aprende a reconocer patrones en los datos.

Una vez que el sistema ha sido entrenado con los datos de entrenamiento, se puede utilizar para hacer predicciones precisas sobre los datos no etiquetados.

Esto te puede resultar muy útil en una gran variedad de aplicaciones del día a día, desde la clasificación de correos electrónicos no deseados hasta la detección de fraudes financieros.

Tipos de Aprendizaje Supervisado.

Existen diferentes tipos de aprendizaje supervisado que se pueden utilizar según las necesidades y objetivos específicos de un proyecto de aprendizaje automático. Cada tipo de aprendizaje supervisado tiene sus propias características y aplicaciones únicas:

Clasificación

Este tipo de aprendizaje supervisado se utiliza para clasificar los datos en diferentes categorías. Por ejemplo, se puede utilizar para clasificar correos electrónicos como spam o no spam.

Regresión
El aprendizaje supervisado de regresión se utiliza para predecir valores numéricos. Por ejemplo, se puede utilizar para predecir el precio de una casa en función de su ubicación y características.

Detección de anomalías
Este tipo de aprendizaje supervisado se usa para identificar patrones anómalos en los datos. Por ejemplo, se puede utilizar para detectar transacciones fraudulentas en una tarjeta de crédito.

Reconocimiento de patrones
El aprendizaje supervisado de reconocimiento de patrones, como bien indica su nombre, se utiliza para identificar patrones en los datos. Por ejemplo, se puede utilizar para reconocer la escritura a mano y convertirla en texto.

Aprendizaje No Supervisado

El **aprendizaje no supervisado** es otra técnica de aprendizaje automático en la que **se utilizan conjuntos de datos no etiquetados y no supervisados para entrenar un modelo.**

A diferencia del aprendizaje supervisado, **en el aprendizaje no supervisado, el modelo no tiene un conjunto de datos de entrenamiento etiquetados y, en su lugar, tiene que identificar patrones y estructuras en los datos por sí solo.**

El objetivo del aprendizaje no supervisado es descubrir patrones ocultos en los datos y agruparlos en categorías o clusters. Algunas de las técnicas más comunes de aprendizaje no supervisado incluyen el clustering, la reducción de dimensionalidad y la detección de anomalías.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Tipos de aprendizaje no supervisado

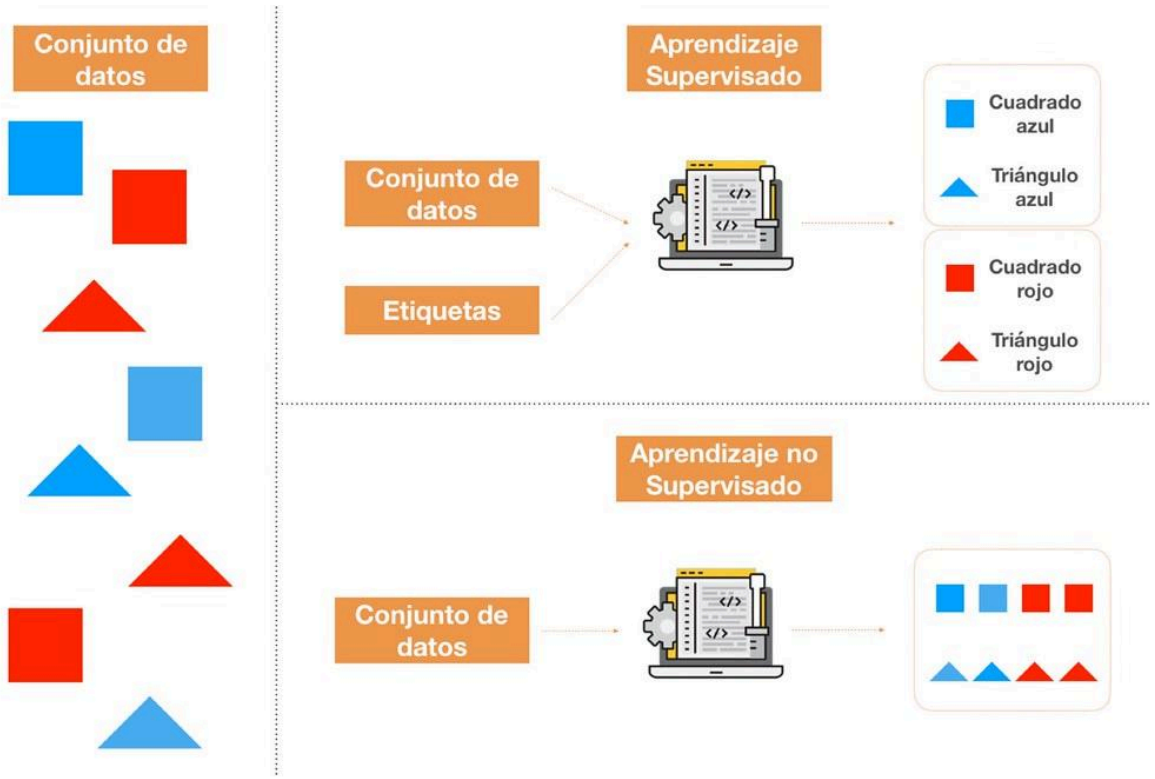
Como ya hemos dejado entrever en el párrafo anterior hay una serie de técnicas utilizadas por este sistema. A continuación algunas de ellas:

Clustering
El clustering se utiliza para agrupar los datos en categorías o clusters basados en sus similitudes y diferencias. Por ejemplo, se puede utilizar para agrupar clientes en diferentes categorías basadas en sus patrones de compra.

Reducción de dimensionalidad
Esta técnica se utiliza para reducir la cantidad de características o variables en un conjunto de datos sin perder información importante. Por ejemplo, se puede utilizar para reducir la cantidad de características en un conjunto de imágenes sin perder información relevante.

Asociación
La asociación es muy útil para descubrir patrones o asociaciones en los datos. Por ejemplo, se puede utilizar para descubrir patrones de compra en un conjunto de transacciones.

Detección de anomalías
Esta técnica se utiliza para identificar patrones anormales o inusuales en los datos. Por ejemplo, se puede utilizar para identificar transacciones fraudulentas en una base de datos de transacciones.



Métricas a evaluar en un modelo de IA.

Las métricas son fundamentales en la evaluación de modelos de IA porque proporcionan una forma objetiva de medir su rendimiento y efectividad en diferentes tareas. Aquí te explico la importancia de cada métrica específica:

1. **Perplejidad (Perplexity):** La perplejidad es crucial en los LLM porque indica qué tan bien el modelo puede predecir palabras o secuencias de palabras basadas en el contexto previo. Una baja perplejidad sugiere que el modelo es capaz de hacer predicciones precisas y seguras, lo cual es esencial para la generación de texto fluido y coherente.

2. **Exactitud (Accuracy):** Aunque la exactitud se aplica más comúnmente en tareas de clasificación, en el contexto de LLM puede referirse a la capacidad del modelo para generar texto gramaticalmente correcto y coherente. Una alta exactitud en este contexto indica que el modelo produce texto que se ajusta adecuadamente al lenguaje y contexto dados.
3. **Coherencia del texto:** Evaluar la coherencia del texto generado es esencial porque un buen modelo de LLM debe ser capaz de producir texto que tenga sentido y siga un flujo lógico y gramatical. Esto es especialmente importante en aplicaciones como la generación de historias, asistentes de texto, entre otros.
4. **Diversidad y creatividad:** La capacidad de un modelo de LLM para generar texto diverso y creativo es importante para evitar la generación repetitiva y fomentar la originalidad en las respuestas. Esto hace que el texto generado sea más interesante y útil en una variedad de aplicaciones.
5. **Tiempo de entrenamiento y eficiencia:** El tiempo de entrenamiento y la eficiencia son críticos en la implementación práctica de modelos de IA. Un modelo eficiente no solo se entrena más rápido, sino que también puede ejecutarse con menos recursos computacionales, lo cual es crucial para su escalabilidad y viabilidad en aplicaciones del mundo real.

En resumen, estas métricas ayudan a los desarrolladores y científicos a evaluar y comparar diferentes modelos de LLM, comprendiendo sus fortalezas y limitaciones. Esto permite mejorar continuamente los modelos para que sean más efectivos en diversas tareas de procesamiento del lenguaje natural y generación de texto automatizada.

¿Qué algoritmos utilizará nuestro ChatBot?

Nuestro Large Language Model (LLM) emplea una combinación de **aprendizaje no supervisado** para el preentrenamiento, **aprendizaje supervisado** para el ajuste fino en tareas específicas, y **aprendizaje profundo** como la base de su arquitectura y funcionamiento.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Configuración de Groq

Para empezar a configurar Groq, **necesitamos nuestra clave secreta que generamos anteriormente en la clase 6**. Esta clave tiene un aspecto similar a esto:

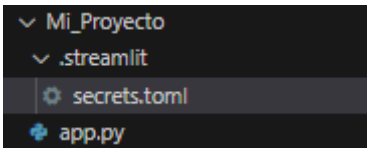
Copygsk_9vjikfjEhlAleS5LNJP8WGdyb3FYmRwUKYsnfyJ9xH4unuXVk0Xo

Es muy importante mantener esta clave en secreto, ¡como si fuera la contraseña de tu instagram!
Si no guardaste la API key, podés generar otra

Guardando la clave secreta en Streamlit

Streamlit nos permite guardar información secreta de forma segura. Para eso, seguí estos pasos:

1. En la carpeta de tu proyecto, crea una nueva carpeta llamada **.streamlit** (no olvides el punto al principio).
2. Dentro de esta nueva carpeta, crea un archivo llamado **secrets.toml**.



3. En este archivo, vamos a guardar nuestra clave API. Escribe lo siguiente:

```
Mi_Proyecto > .streamlit > secrets.toml
1   CLAVE_API = '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'
2
```

Reemplaza la clave con tu clave real.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

🚨 **¡Atención! El archivo secrets.toml debe mantenerse secreto. No lo compartas ni lo subas a repositorios públicos como GitHub. Si estás usando una computadora compartida, borra la clave antes de irte.**

Creación de un usuario de Groq

Ahora que tenemos configurado el acceso a Groq, vamos a crear una función para generar un usuario que usaremos en cada mensaje que enviemos a nuestro chatbot.

Importando la librería de Groq
Primero, importamos la librería de Groq que instalamos anteriormente. Al principio de tu archivo Python, después de importar Streamlit, añade esta línea:

pythonCopyfrom groq import Groq

Creando la función para el usuario de Groq

Ahora, vamos a crear una función que nos ayude a conectarnos con Groq:

```
def crear_usuario_groq():
    clave_secreta = st.secrets["CLAVE_API"]
    return Groq(api_key=clave_secreta)
```

Explicación:

1. **st.secrets["CLAVE_API"] :** lee nuestra clave secreta del archivo secrets.toml.
2. Luego, usamos esta clave para crear y devolver un usuario de Groq.

Esta función la usaremos cada vez que necesitemos enviar un mensaje a nuestro chatbot.

Configuración del modelo y manejo de mensajes

Ahora que tenemos nuestro usuario de Groq, vamos a crear algunas funciones para configurar el modelo y manejar los mensajes, por el momento solo de entrada.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Configurando el modelo

Antes de configurar nuestro modelo, será necesario decirle al usuario que modelos tiene Groq para ofrecernos. Para eso, puedo ir a la página de groq, copiar los modelos disponibles y reemplazar los modelos creados anteriormente en la variable MODELOS.

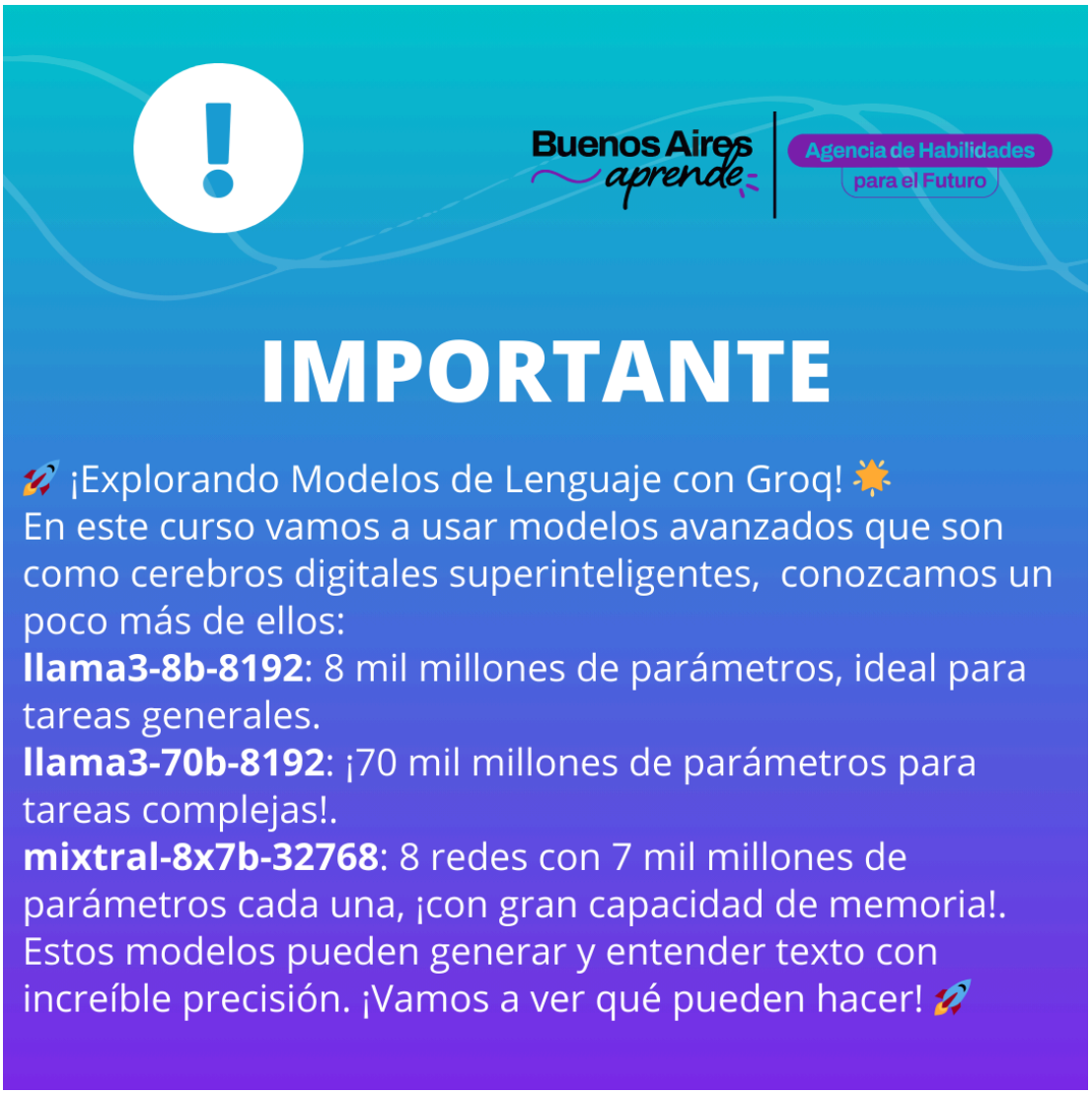
Pasamos de esto:

```
MODELOS = ['modelo1', 'modelo2', 'modelo3'] # Clase 6
```

a esto:

```
MODELOS = ['llama3-8b-8192', 'llama3-70b-8192', 'mixtral-8x7b-32768']
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro



🚨 **¡Atención! Es importante que la variable MODELO esté por encima de la función configurar_pagina(). Esto se debe a que la función, usa la variable MODELOS en sus parámetros, por lo tanto al momento de crearse la función, tiene que existir dicha variable.**

Primero, vamos a crear una función para configurar nuestro modelo de chat:

```python
def configurar_modelo(cliente, modelo, mensajeDeEntrada):
    return cliente.chat.completions.create(
        model=modelo,
        messages=[{"role": "user", "content": mensajeDeEntrada}],
        stream=True
    )
```

Esta función hace lo siguiente:

- Toma **tres parámetros:**
  - Cliente que va a ser nuestro usuario de Groq, creado anteriormente.
  - Es el nombre del modelo de IA que queremos usar. Groq tiene diferentes modelos, como diferentes "cerebros" para diferentes tareas.
  - mensajeDeEntrada: Este es el mensaje que el usuario escribe en nuestro chatbot. Es lo que queremos que el modelo de IA lea y responda.
- **Return:** Como lo que espero es únicamente una respuesta, crearemos un return que será una "completion" de chat, que es básicamente una respuesta del modelo.
  - **model=modelo**: Aquí especificamos qué "cerebro" o modelo de IA queremos usar.Usamos el parámetro modelo que pasamos a la función
  - **messages=[{"role": "user", "content": mensajeDeEntrada}]:** Esto es una lista que contiene el mensaje del usuario.
    - **"role": "user"** indica que este mensaje viene del usuario, no del chatbot. otras opciones son:
      - **"assistant"**: Este rol se utiliza para los mensajes generados por el modelo de lenguaje o chatbot. Representa las respuestas del asistente virtual.
      - **"system":** Este rol se usa para mensajes que proporcionan contexto o instrucciones al modelo. Estos mensajes no son parte de la conversación visible para el usuario, sino que sirven para configurar el comportamiento del modelo.
      - **"function"**: Aunque menos común, algunos sistemas utilizan este rol para representar la salida de una función llamada durante la conversación.
    - **"content": mensajeDeEntrada** es el contenido real del mensaje que escribió el usuario.

- **Stream=True** : Este parámetro del modelo lo configura para que responda en tiempo real. Es como si el chatbot estuviera "pensando en voz alta" mientras escribe su respuesta.

Cache e inicio del estado de la sesión

En **Streamlit**, necesitamos una forma de recordar los mensajes anteriores. Para eso, usamos algo llamado "estado de sesión". Vamos a crear una función para inicializar este estado de sesión:

```
def inicializar_estado():
    if "mensajes" not in st.session_state:
        st.session_state.mensajes = []
```

¿Qué hace esta función?

- Comprueba si ya tenemos una lista de mensajes guardada en **st.session_state.**
- Si no la tenemos, crea una lista vacía para guardar nuestros mensajes.

El estado de sesión es como una memoria especial de Streamlit. Nos permite guardar información entre diferentes interacciones del usuario con nuestra aplicación. Es muy útil para aplicaciones de chat, porque nos permite recordar toda la conversación.

¿Es una memoria Cache?

Se parece mucho a una memoria caché, porque el estado de sesión guarda información temporalmente,hasta que la sesión se cierre, pero de una manera más controlada y accesible que una caché típica. Mientras se mantenga abierta la aplicación en tu navegador, la información se mantiene, incluso si se recarga la página. Esta memoria puede guardar cientos de mensajes sin problema, pero no es ilimitada

¿Por qué es útil para nuestro chatbot?

- **Memoria de la conversación**: Nos permite recordar todos los mensajes intercambiados durante la sesión actual. Así, el chatbot puede "recordar" lo que se ha dicho antes y mantener una conversación coherente.

- **Personalización**: Podemos guardar preferencias del usuario durante la sesión, como el modelo de IA que prefiere usar o parámetros del promp-engineer.
- **Experiencia fluida:** Aunque recargues la página, la conversación no se pierde, lo que hace que la experiencia de uso sea más agradable.

Usando nuestras funciones

Ahora que tenemos nuestras funciones, vamos a usarlas:

```
clienteUsuario = crear_usuario_groq()
inicializar_estado()

# Tomamos el mensaje del usuario por el input.
mensaje = st.chat_input("Escribí tu mensaje:")

# Verificamos que el mensaje no esté vacío antes de configurar el modelo
if mensaje:
    configurar_modelo(clienteUsuario, elegirModelo, mensaje)
    print(mensaje)  # Mostramos el mensaje en la terminal para ver cómo
se muestra
```

Explicación paso a paso:

1. Creamos nuestro usuario de Groq y lo guardo en una variable llamada **clienteUsuario**.
2. Inicializamos el estado de la sesión para guardar los mensajes. “Activamos nuestra memoria caché”
3. Creo una variable llamada **mensaje** con la función **st.chat_input()** que nos va a permitir crear un campo donde el usuario puede escribir su mensaje, y comenzar el historial.
4. Usamos los condicionales para guiar nuestra aplicación. Verificamos si el usuario escribió algo:
    - Si escribió un mensaje, configuramos el modelo con ese mensaje.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Próximos pasos

Con esta configuración básica, ya estamos listos para empezar a crear nuestro chatbot. En las siguientes secciones, aprenderemos cómo:

1. Agregar y mostrar el historial a la interfaz del chatbot con Streamlit
2. Enviar mensajes al modelo de Groq
3. Recibir y mostrar las respuestas del chatbot

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Desafío N° 7:

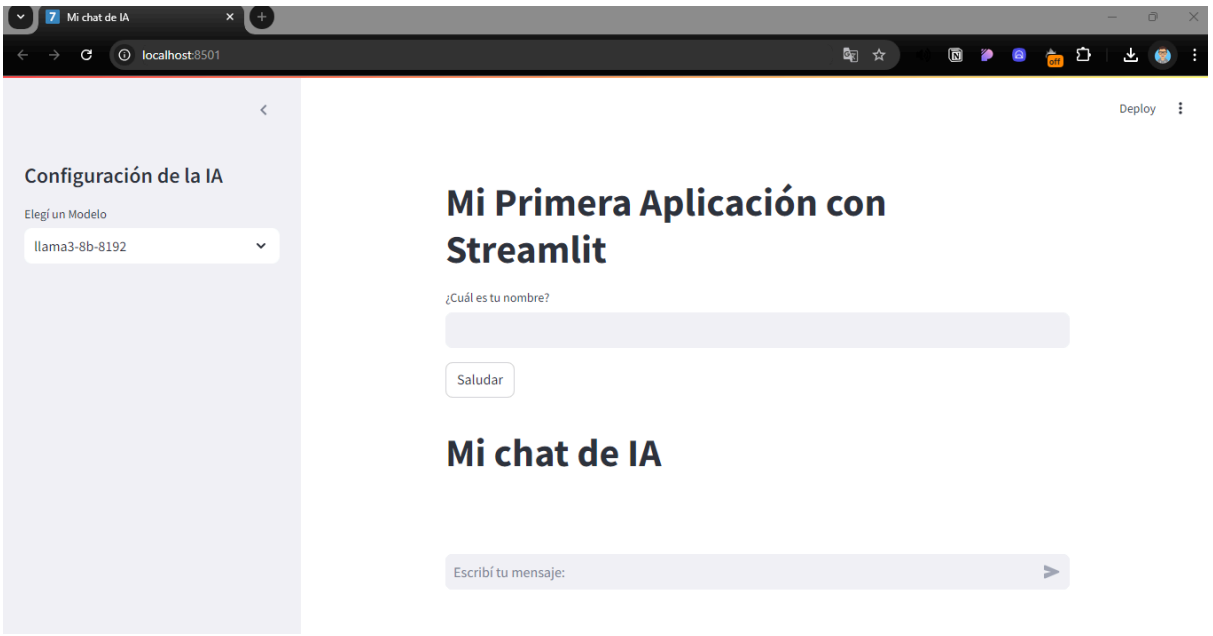
En este desafío vamos a entregar lo visto en clase hasta el momento. **Recordá no enviar la API Key o eliminarla de nuestro proyecto una vez que la usemos**. **Te pedimos que nos envíes dos capturas de pantalla:**

1. Imagen del navegador con tu aplicación corriendo, a la que le enviaste un mensaje
2. Imágen de la salida por terminal que te devuelve la app.

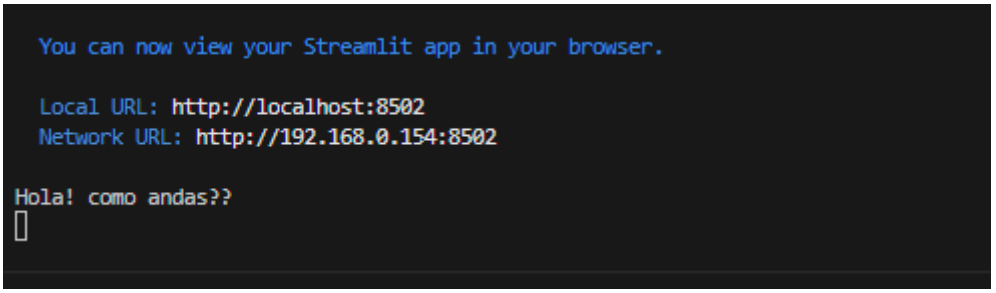
Mínimamente el usuario tiene que poder:

1. Elegir uno de los modelos modelos de groq.
2. Mandar un mensaje por el **chat_input** que salga por la terminal.

El proyecto debe verse más o menos así:



Y cuando envío el mensaje por el navegador, en mi terminal tiene que verse así!



Los números en Local URL y Network URL pueden variar


Buenos Aires
aprende
Buenos
Aires
Ciudad