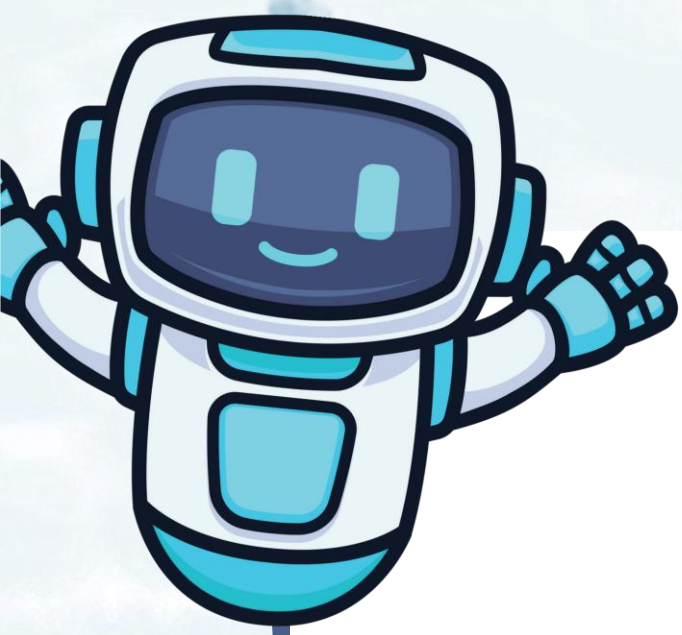
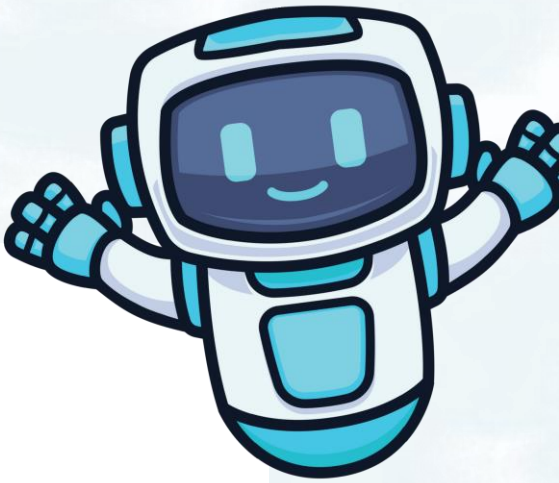
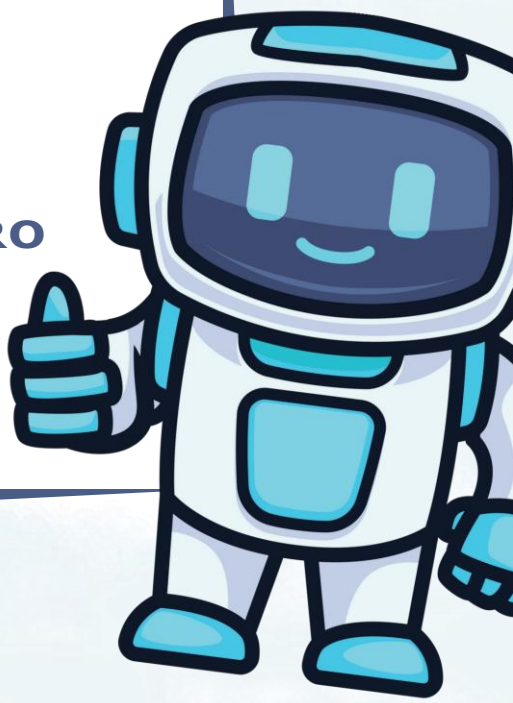




INTELIGENCIA ARTIFICIAL EN TU PROPIO PC

CARMEN FELIPE NAVARRO



ÍNDICE

1. INVESTIGACIÓN Y SELECCIÓN

En los últimos años, la inteligencia artificial (IA) ha avanzado de manera notable, impulsada por modelos sofisticados que pueden procesar enormes cantidades de datos y realizar tareas complejas. Estos modelos, que se basan principalmente en redes neuronales profundas, se han implementado con éxito en diversas áreas, como el **procesamiento del lenguaje natural** (NLP), la visión por computadora y el análisis de grandes volúmenes de datos.

En particular, los **Modelos Fundacionales** han adquirido gran relevancia, ya que funcionan como la base sobre la que se desarrollan muchas aplicaciones innovadoras en IA. Un subconjunto clave dentro de estos modelos son los **Modelos de Lenguaje de Gran Tamaño** (LLM, por sus siglas en inglés), diseñados específicamente para comprender y generar texto con alta precisión, lo que los hace esenciales para tareas de NLP, como la traducción automática, el análisis de sentimientos, la generación de texto y la respuesta a preguntas.

Teniendo en cuenta este panorama, he investigado diversos modelos de IA, profundizando en cómo cada uno de ellos contribuye al avance de las aplicaciones actuales y futuras.

1.1 Investigación sobre diversos modelos de IA

i. GPT-3

GPT-3 (Generative Pre-trained Transformer 3) es un modelo de inteligencia artificial desarrollado por OpenAI en 2020. Con 175 mil millones de parámetros, es capaz de generar textos coherentes y relevantes en diversos contextos. Su entrenamiento se basa en una gran cantidad de datos de Internet, lo que le permite responder preguntas, escribir ensayos, traducir idiomas y generar código. GPT-3 es ampliamente utilizado en chatbots, asistentes virtuales y herramientas de generación de contenido. Su capacidad para generar respuestas coherentes lo hizo destacar en el campo del procesamiento del lenguaje natural.

ii. Jurassic-1

Jurassic-1 creado por AI21 Labs en 2021, es un modelo con **178 mil millones de parámetros**, diseñado como una alternativa a GPT-3 que se especializa en la generación de textos naturales y en la reescritura de contenido. Su enfoque se centra en ofrecer herramientas de IA accesibles para desarrolladores y empresas que buscan integrar generación de lenguaje en sus aplicaciones. Es utilizado en redacción avanzada, generación de artículos y asistencia en escritura.

iii. Megatron- Turing NGI

Desarrollado por NVIDIA y Microsoft, es un modelo altamente escalable con versiones que llegan hasta **530 mil millones de parámetros** en su variante **Megatron-Turing NLG**. Está diseñado para entrenarse en supercomputadoras y mejorar la eficiencia del procesamiento de lenguaje natural, siendo utilizado en análisis de datos, generación de texto y traducción automática.

iv. Bard

Bard (Gemini) es el modelo desarrollado por Google DeepMind, basado en **PaLM 2** y ahora en la serie **Gemini**. Es un modelo multimodal, lo que significa que puede procesar no solo texto, sino también imágenes y código. Su principal aplicación es mejorar las interacciones conversacionales y la búsqueda de información en tiempo real.

v. ChatGP

ChatGPT es un chatbot basado en la familia de modelos GPT de OpenAI. Desde su lanzamiento en 2022, ha evolucionado hasta incluir versiones como GPT-3.5 y GPT-4. Su capacidad para comprender y generar texto en lenguaje natural lo ha convertido en una herramienta popular en educación, atención al cliente y generación de contenido.

vi. BLOOM – BigScience

Desarrollado por más de 1.000 investigadores voluntarios en colaboración con el startup de IA Abragging Face, BLOOM es una LLM entrenada con 176 mil millones de parámetros durante 117 días en el Centro Nacional de Investigación Científica de Francia.

BLOOM tiene un fuerte enfoque en la transparencia, con los desarrolladores compartiendo abiertamente los datos que están utilizando para entrenar el modelo, los desafíos de desarrollar el modelo y cómo el grupo evalúa el rendimiento del modelo. Esto da a los forasteros una visión más profunda de cómo funciona BLOOM, convirtiéndola en una de las LLM más transparentes. Aparte de la ética, BLOOM está en vivo ahora y es libre de usar, y los usuarios pueden elegir entre una gama de modelos de idiomas para adaptarse a sus necesidades y requerimientos de hardware.

vii. LLaMA (Large Language Model Meta AI)

Es un modelo creado por Meta, con variantes desde **7B hasta 65B parámetros**. Fue desarrollado con el objetivo de ofrecer un modelo más eficiente y accesible para la comunidad de investigación en inteligencia artificial. Se utiliza en procesamiento del lenguaje natural, generación de contenido y análisis de datos.

viii. Mistral 7B

Lanzado en 2023 por Mistral AI, es un modelo compacto con **7 mil millones de parámetros**. A pesar de su menor tamaño, su rendimiento es comparable al de modelos más grandes. Su principal ventaja es la eficiencia computacional, lo que lo hace ideal para aplicaciones que requieren IA potente con **menos consumo de recursos.**

ix. Falcon LLM

Desarrollado por el Instituto de Innovación Tecnológica de los Emiratos Árabes Unidos, es un modelo disponible en versiones de **7B y 40B parámetros**. Se ha destacado por su alto rendimiento y por ser una alternativa de código abierto en el campo del procesamiento del lenguaje natural, utilizado en chatbots y generación de contenido.

x. **GPT-Neo y GPT-J (EleutherAI)**

Desarrollado por EleutherAI, es una alternativa de código abierto a GPT-3, con versiones de **1.3B y 2.7B parámetros**. Su objetivo es ofrecer una opción accesible para investigadores y desarrolladores interesados en la generación de texto y modelos de lenguaje sin depender de plataformas comerciales cerradas.

xi. **Vicuna**

Es un modelo basado en **LLaMA**, optimizado para mejorar la interacción conversacional. Se ha entrenado con datos de interacciones humanas para generar respuestas más naturales y coherentes, lo que lo hace adecuado para aplicaciones como asistentes virtuales y chatbots avanzados.

xii. **Perpleja AI**

Es un modelo diseñado específicamente para mejorar la precisión de las respuestas generadas por la inteligencia artificial. Su principal objetivo es reducir la generación de información errónea, lo que lo convierte en una opción confiable para asistentes virtuales y sistemas de búsqueda inteligente.

xiii. **Bert**

Es uno de los modelos más importantes en el campo del procesamiento de lenguaje natural (NLP). Desarrollado por Google en 2018, BERT ha revolucionado el modo en que las máquinas entienden el lenguaje, siendo capaz de captar el contexto de una palabra en relación con las palabras que la rodean, algo que no era posible en modelos anteriores.

xiv. **Bart (Bidirectional and Auto-Regressive Transformers)**

Es un modelo de lenguaje desarrollado por **Facebook AI** que combina características de modelos bidireccionales (como BERT) y autorregresivos (como GPT). Esta combinación le permite ser altamente eficiente para tareas de procesamiento de lenguaje natural (NLP), como resumen de texto, traducción automática, y generación de texto.

1.2 Justificación de la Elección de Bart-small

Inicialmente, consideré el modelo BART debido a su versatilidad y alto rendimiento en tareas de procesamiento del lenguaje natural (NLP). Su combinación de capacidades bidireccionales y autorregresivas lo hace ideal para tareas como resumen, generación de contenido y traducción automática. Sin embargo, al evaluar los requisitos de hardware, me di cuenta de que BART es un modelo exigente en términos de memoria y potencia de procesamiento, lo que podría afectar al rendimiento de mi ordenador por los recursos utilizados. Por esta razón, opté por BART-small, una versión más ligera y optimizada que permite un uso eficiente sin comprometer demasiado la calidad de los resultados.

Además de los factores mencionados, la elección de BART-small también se basa en consideraciones sobre la escalabilidad y eficiencia de los recursos. BART-small permite realizar una ejecución más rápida

y un menor consumo de memoria, lo cual es esencial para proyectos con presupuestos limitados en términos de infraestructura o cuando el entorno de trabajo no dispone de recursos de hardware de alto rendimiento. Al optar por una versión optimizada, se reduce el riesgo de posibles cuellos de botella que podrían afectar a otros procesos o aplicaciones que estén ejecutándose simultáneamente.

No obstante, debido a la falta de una implementación oficial de BART-small en Hugging Face, las pruebas funcionales se han realizado utilizando BART (modelo base o grande). Aunque este modelo es más grande, su arquitectura y funcionamiento son equivalentes a los de BART-small, por lo que los resultados obtenidos siguen siendo representativos y válidos para el análisis de funcionalidades.

Además, las pruebas realizadas con el modelo BART-base siguen siendo relevantes ya que permiten validar el comportamiento y la versatilidad de la arquitectura de BART en general. Las diferencias de rendimiento entre las versiones no deberían impactar negativamente los resultados de las pruebas funcionales si los objetivos son evaluar las capacidades del modelo en tareas específicas, como el resumen, la generación de contenido o la traducción automática. De hecho, realizar las pruebas con BART-base (como sustituto de BART-small) ofrece una perspectiva adicional sobre el desempeño del modelo bajo condiciones de mayor carga computacional, lo que también puede proporcionar valiosos datos sobre la robustez y escalabilidad del enfoque.

Esta elección me ha permitido mantener el equilibrio entre un modelo adecuado para mi entorno de trabajo y la posibilidad de realizar pruebas completas sin comprometer la validez de los resultados. En conclusión, la elección de BART-small se alinea perfectamente con los requisitos técnicos y operacionales del proyecto, mientras que el uso de modelos más grandes como BART-base garantiza que las pruebas sean completas y ofrezcan una visión integral de las capacidades del modelo en diversos escenarios.

Característica	BART	BART-small
Tamaño	Grande	Pequeño
Uso principal	Resumen, Generación de texto	Resumen, Generación de texto
Rendimiento	Alto (requiere mucha RAM y CPU)	Menor rendimiento, pero rápido
Requisitos de hardware	Alta demanda de recursos (GPU recomendada)	Requiere menos recursos, adecuado para PCs de gama media
Aplicaciones	Traducción, Resumen, Generación	Resumen, Generación de contenido

1.3 Información recopilada

Hugging Face (https://huggingface.co/docs/transformers/model_doc/bart) dice que BART es un modelo desarrollado por Facebook AI que utiliza una arquitectura de codificador-decodificador, combinando características de modelos bidireccionales y autorregresivos. Es eficaz en tareas de generación de texto, resumen, traducción y otras aplicaciones de procesamiento del lenguaje natural (NLP). Esta fuente explica que BART se entrena corrompiendo el texto de entrada y aprendiendo a reconstruirlo, lo que mejora su capacidad de generar texto coherente.

Según Analytics Vidhya (https://www.analyticsvidhya.com/blog/2024/11/bart-model/), se ha demostrado que BART supera a otros modelos en tareas como la generación de respuestas y el resumen automático, aunque requiere una gran cantidad de datos y poder computacional para su implementación.

Por otro lado, ProjectPro (https://www.projectpro.io/recipes/what-is-bart-model-transformers) compara BART con modelos como T5 y GPT, destacando su flexibilidad en tareas de NLP. A diferencia de GPT, que es solo un modelo generativo, BART combina capacidades bidireccionales para mejorar la comprensión del contexto.

Según Inteligencia Artificial 360 (https://inteligenciaartificial360.com/fundamentos-ia/modelos-de-lenguaje/t5-y-bart-modelos-de-lenguaje-secuencia-a-secuencia-y-tareas-de-generacion/) , BART se ha utilizado en tareas como la generación de respuestas automáticas y la reformulación de preguntas. Esta fuente menciona que sus principales ventajas incluyen una mayor coherencia en la generación de texto y una mejor comprensión del contexto, pero también advierte sobre sus limitaciones, como la posibilidad de generar respuestas incorrectas o sesgadas.

Growketing dice que (https://www.growketing.com/ia-generativa-como-transformers-bert-y-gpt-estan-cambiando-el-marketing-digital/) , los modelos de IA generativa, incluidos BART, BERT y GPT, están revolucionando el marketing digital. Se utilizan para automatizar la generación de contenido, mejorar la personalización y optimizar la interacción con clientes. Esta fuente destaca que uno de los principales desafíos es evitar sesgos en los modelos y garantizar la calidad del contenido generado.

Según Las Cosas de Internet (https://lascosasdeinternet.com/tech-development/inteligencia-artificial/google-gemini-la-ia-de-google-para-2024/) , Google ha lanzado Gemini, su nuevo modelo de IA para 2024. Esta fuente destaca que Gemini se diferencia por su capacidad multimodal, que le permite trabajar con texto, imagen y audio. Además, tiene distintas versiones (Ultra, Pro y Nano) adaptadas a diferentes usos, desde dispositivos móviles hasta soluciones empresariales.

1.3.1 Curiosidades

¿Por qué "facebook/bart-small"?

facebook/ → Significa que el modelo fue entrenado y publicado por **Facebook AI** en Hugging Face.
bart-small → Sería una versión reducida del modelo BART (Bidirectional and Auto-Regressive Transformer), aunque no es un modelo oficial.
En Hugging Face, los modelos oficiales de BART incluyen:

- facebook/bart-base
- facebook/bart-large
- facebook/bart-large-cnn (optimizado para resúmenes)
- facebook/bart-large-mnli (para clasificación de texto)

Meta (Facebook) ha desarrollado varios modelos de IA famosos, como:

- **LLaMA** (para procesamiento de lenguaje natural avanzado)
- **Detectron2** (para visión por computadora)
- **wav2vec** (para reconocimiento de voz)

Por lo tanto, el nombre **"facebook/bart-small"** simplemente indica que fue desarrollado por Meta (Facebook AI), pero no significa que el modelo esté vinculado directamente con la red social Facebook.

Pre-entrenamiento

Los hiperparámetros de entrenamiento se configuran para aprovechar al máximo el rendimiento de hardware disponible. El modelo se entrena utilizando **8 GPUs A100 con precisión FP32** optimizada, lo que permite una ejecución eficiente y rápida. El tamaño total del lote es de **1024 muestras**, lo que asegura que se procesen grandes cantidades de datos de manera simultánea. El entrenamiento consta de 200,000 pasos, y la **longitud máxima** de las **secuencias** es de **512 tokens**, permitiendo un análisis y procesamiento extenso de cada entrada.

En cuanto al preprocesamiento, se utiliza un enfoque de "**denoising**" para mejorar la calidad de los datos. La probabilidad de aplicar esta técnica es del 30%, con un máximo de 200 segmentos modificados por muestra. La estrategia de enmascarado se asemeja a la de BERT, donde se enmascaran palabras completas, y la distribución de longitud de las palabras a eliminar sigue una **distribución Poisson** con un valor promedio de 2.5 palabras. Además, se emplea una técnica de permutación de frases para variar la estructura de las entradas y ayudar al modelo a aprender representaciones más robustas.

Para la **optimización**, se emplea el **algoritmo AdamW**, que ha demostrado ser eficiente en tareas de procesamiento de lenguaje natural. La tasa de aprendizaje sigue un esquema triangular con un pico en 1e-4, permitiendo un ajuste dinámico durante el entrenamiento. Se realizan 10,000 pasos de calentamiento para estabilizar la tasa de aprendizaje al inicio del entrenamiento, y se aplica un **decaimiento** de **pesos** con un valor de **0.01** para **evitar** el **sobreajuste** y garantizar la generalización del modelo.

Los **conjuntos** de **datos** utilizados para el entrenamiento son diversos y abarcan una amplia gama de fuentes de texto. Se incluyen el BookCorpus, CC-News, OpenWebText y la **Wikipedia** en

inglés, lo que asegura que el modelo pueda aprender de un contenido variado y representativo de múltiples dominios. Estos conjuntos proporcionan textos de diferentes géneros y contextos, lo que permite entrenar un modelo capaz de comprender y generar texto de alta calidad en diversas situaciones.

1.4 Ventajas y Desventajas

Punto de vista personal

BART (modelo completo) ofrece grandes ventajas para tareas complejas. Es ideal para usuarios que necesitan generar texto de alta calidad, como resúmenes detallados, traducciones o análisis de sentimientos. Su flexibilidad permite realizar tareas variadas, y su capacidad para manejar datos complejos lo convierte en una herramienta poderosa. Sin embargo, tiene desventajas, como la necesidad de recursos computacionales elevados, lo que puede no ser accesible para todos los usuarios. Además, la generación de resultados puede ser más lenta debido a su tamaño y complejidad, lo que podría resultar frustrante en situaciones que requieren respuestas rápidas. También, su uso puede generar dependencia de tecnología avanzada.

Por otro lado, **BART-small** es una versión más ligera y rápida, lo que lo hace más accesible para usuarios con equipos menos potentes. Es perfecto para tareas cotidianas y simples, como generar respuestas rápidas o asistencia en tareas personales. La facilidad de uso también es una ventaja importante, ya que no requiere conocimientos avanzados para integrarlo. Sin embargo, tiene limitaciones en tareas más complejas, como la creación de contenido extenso o el análisis profundo de texto. En estas situaciones, los resultados pueden no ser tan coherentes ni detallados como los de la versión completa.

Punto de vista empresarial

Desde una perspectiva empresarial, **BART** (modelo completo) es ideal para empresas que necesitan un alto rendimiento en tareas complejas, como la generación de contenido detallado, el análisis de grandes volúmenes de datos o la mejora de la atención al cliente. Su capacidad para personalizarse mediante el **fine-tuning** permite adaptarlo a las necesidades específicas de cada empresa, ofreciendo soluciones precisas. No obstante, la implementación de BART requiere una infraestructura robusta y costosa, además de un alto consumo energético, lo que puede aumentar significativamente los costos operativos. La complejidad en su integración también puede ser un reto para algunas empresas.

Por su parte, **BART-small** es una opción más económica y rápida. Su menor consumo de recursos lo hace ideal para empresas con presupuesto limitado o que necesiten soluciones escalables sin incurrir en grandes costos. Es adecuado para tareas específicas y de menor complejidad, como generar respuestas automáticas o manejar interacciones básicas con los clientes. Sin embargo, al ser más limitado en capacidad, no es adecuado para tareas que requieren un análisis profundo o la creación de contenido más detallado. Las empresas que necesiten adaptarse rápidamente a cambios complejos en los datos podrían encontrar que **BART-small** no ofrece la flexibilidad necesaria.

2. INSTALACIÓN y PRUEBAS

2.1 Instalación

1. Crear un entorno virtual

Un entorno virtual te permite instalar paquetes de Python de forma aislada, sin afectar otras configuraciones o proyectos en tu sistema.

1.1. Navega hasta la carpeta donde deseas crear el entorno virtual y ejecuta el comando para crear el entorno: **py -m venv bart_env** (Esto generará una carpeta llamada *bart_env* dentro del proyecto).

```
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small> py -m venv bart_env
>>
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small>
```

1.2. Activar el entorno visual: **bart_env\Scripts\activate**

```
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small> bart_env\Scripts\activate
>>
bart_env\Scripts\activate : No se puede cargar el archivo C:\Users\USUARIA\Documents\Master\MIA\2
trimestre\bart-small\bart_env\Scripts\Activate.ps1 porque la ejecución de scripts está deshabilitada en este sistema. Para obtener más
información, consulta el tema about_Execution_Policies en https:/go.microsoft.com/fwlink/?LinkID=135170.
En línea: 1 Carácter: 1
+ bart_env\Scripts\activate
+ ~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small>
```

Este error se debe a que **PowerShell tiene bloqueada la ejecución de scripts** por razones de seguridad.

Para poder solucionar el problema:

i. Abrir CMD
ii. Ve a la carpeta de tu proyecto**:**
iii. Activa el entorno virtual con: *bart_env\Scripts\activate.bat*

¿Por qué pasa esto?

Windows tiene restricciones de seguridad que bloquean la ejecución de scripts .ps1 en PowerShell. El comando *Set-ExecutionPolicy Unrestricted -Scope Process* solo cambia la configuración temporalmente para la sesión actua**l**, por lo que cada vez que cierres PowerShell, la restricción volverá.

¿Cómo saber cuando el entorno virtual esta activado?

Después de activarlo, la terminal mostrará algo como esto: *(bart_env) user@machine:~$*

El prefijo (bart_env) indica que estás dentro del entorno virtual.

```
Microsoft Windows [Versión 10.0.22631.4602]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\USUARIA>cd "C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small"

C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small>bart_env\Scripts\activate.bat

(bart_env) C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small>
```

```
Administrador: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\WINDOWS\system32> Set-ExecutionPolicy Unrestricted -Scope Process
>>

Cambio de directiva de ejecución
La directiva de ejecución te ayuda a protegerte de scripts en los que no confías. Si cambias dicha directiva, podrías
exponerte a los riesgos de seguridad descritos en el tema de la Ayuda about_Execution_Policies en
https:/go.microsoft.com/fwlink/?LinkID=135170. ¿Quieres cambiar la directiva de ejecución?
[S] Sí  [O] Sí a todo  [N] No  [T] No a todo  [U] Suspender  [?] Ayuda (el valor predeterminado es "N"): S
PS C:\WINDOWS\system32>
```

2. Instalar paquetes dentro del entorno virtual

Cuando el entorno está activado, los paquetes que instales con pip se guardarán solo dentro de bart_env, sin afectar el Python global de tu sistema. Por ejemplo: *pip install transformers torch*

```
(bart_env) C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small> pip install transformers torch
Collecting transformers
  Downloading transformers-4.48.0-py3-none-any.whl.metadata (44 kB)
Downloading mpmath-1.3.0-py3-none-any.whl (536 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 536.2/536.2 kB 2.2 MB/s eta 0:00:00
Using cached urllib3-2.3.0-py3-none-any.whl (128 kB)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: mpmath, urllib3, typing-extensions, sympy, setuptools, safetensors, regex, pyyaml, packag
ing, numpy, networkx, MarkupSafe, idna, fsspec, filelock, colorama, charset-normalizer, certifi, tqdm, requests, jinja2,
 torch, huggingface-hub, tokenizers, transformers

[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

3. Descargar e Implementar BART-Small

3.1. Método 1: Cargar un Modelo Similar (facebook/bart-base)

Dado que no existe facebook/bart-small en Hugging Face, se puede usar facebook/bart-base, que es un modelo más pequeño que bart-large.

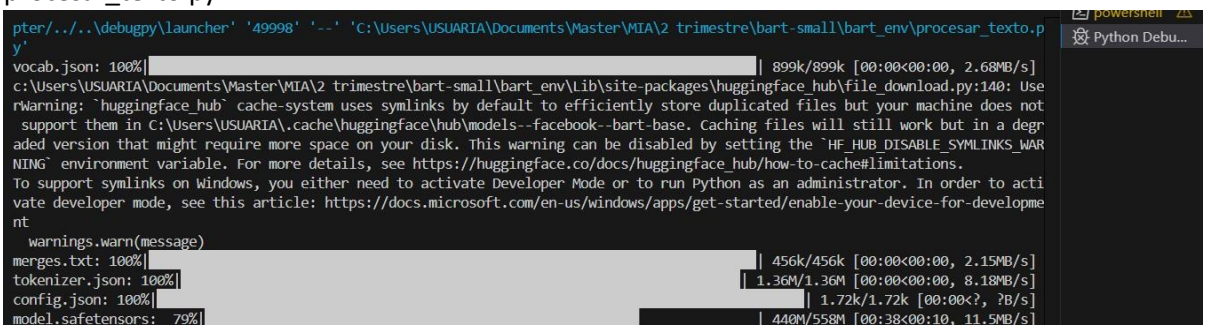
i. Crea un archivo de Python (script)
En tu carpeta de proyecto, crea un archivo Python, por ejemplo: *procesar_texto.py*

ii. Escribe o copia el código dentro del archivo *procesar_texto.py*
from transformers import BartTokenizer, BartForConditionalGeneration

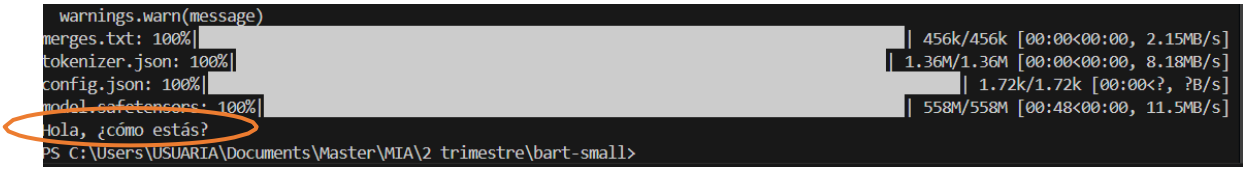
```
# Modelo base de BART (más pequeño que Large)
model_name = "facebook/bart-base"
tokenizer = BartTokenizer.from_pretrained(model_name)
model = BartForConditionalGeneration.from_pretrained(model_name)
```

```
# Prueba de tokenización
text = "Hola, ¿cómo estás?"
inputs = tokenizer(text, return_tensors="pt")
outputs = model.generate(**inputs)

# Decodificar la salida
print(tokenizer.decode(outputs[0],  skip_special_tokens=True))
```

iii. Ejecuta el script en tu terminal o en VS Code mediante el comando: py procesar_texto.py



Este código cargará el modelo *facebook/bart-base*, lo tokenizará, generará una salida y luego decodificará el texto. En este caso, el texto de entrada es "Hola, ¿cómo estás?", y la salida será el texto generado por el modelo.



3.2. Metodo 2: Crear y Entrenar un BART-Small Personalizado

Para tener un modelo más pequeño, puedes reducir manualmente los parámetros de bart-base y ajustarlo con un dataset propio.

```
from transformers import BartConfig, BartForConditionalGeneration

# Configurar un modelo más pequeño
config = BartConfig(
   vocab_size=50265,
   encoder_layers=4,  # Reducido de 12 en `bart-base`
   decoder_layers=4,
   d_model=256,       # Dimensión reducida (vs 768 en `bart-base`)
   encoder_attention_heads=4,
   decoder_attention_heads=4
)

# Crear el modelo personalizado
small_bart = BartForConditionalGeneration(config)

print("Modelo BART-Small creado con éxito.")
```

4. Guardar el Modelo para Uso Offline

4.1. Guardar el modelo y el Tokenizador en local

El código mostrado a continuación descarga el modelo y el tokenizador desde *Abrazo Face* (requiere conexión a Internet) y guarda los archivos necesarios en una carpeta local llamada *bart-base-local*.

```
from transformers import BartTokenizer, BartForConditionalGeneration

# Nombre del modelo de Hugging Face
model_name = "facebook/bart-base"

# Descargar el modelo y el tokenizador
tokenizer = BartTokenizer.from_pretrained(model_name)
model = BartForConditionalGeneration.from_pretrained(model_name)

# Guardar el modelo y el tokenizador en la carpeta local "./bart-base-local"
model.save_pretrained("./bart-base-local")
tokenizer.save_pretrained("./bart-base-local")

print("Modelo y tokenizador guardados en la carpeta './bart-base-local'")
```

El mensaje indica que el modelo y el tokenizador se han guardado en la carpeta *./bart-base-local*. El aviso adicional sobre la configuración de la generación no afecta a la veda simplemente e informando ciertos parámetros se están moviendo de una configuración a otro internamente en la biblioteca *transformers*.

```
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small> & "C:/Users/USUARIA/Documents/Master/MIA/2 trimestre/bart-small/
bart_env/Scripts/python.exe" "c:/Users/USUARIA/Documents/Master/MIA/2 trimestre/bart-small/bart_env/procesar_texto.py"
C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small\bart_env\Lib\site-packages\transformers\modeling_utils.py:2758: User
Warning: Moving the following attributes in the config to the generation config: {'early_stopping': True, 'num_beams': 4, 'no_rep
eat_ngram_size': 3, 'forced_bos_token_id': 0}. You are seeing this warning because you've set generation parameters in the model
config, as opposed to in the generation config.
  warnings.warn(
Modelo y tokenizador guardados en la carpeta './bart-base-local'
```

4.2. Modificar el Código para Usar el Modelo Offline

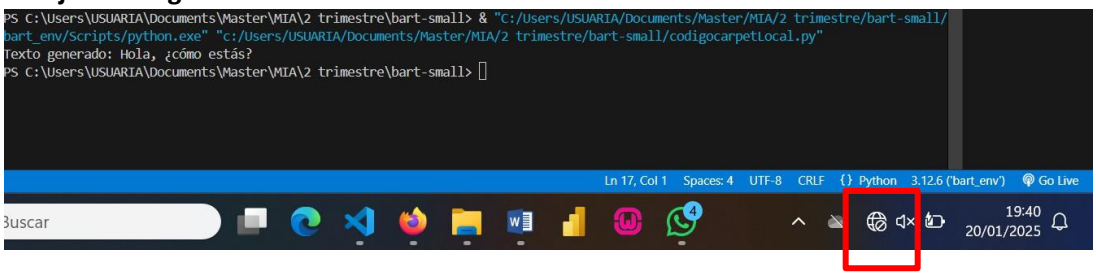
Ahora que el modelo y el tokenizador están guardados localmente, hay que desconectarse de Internet y probar el modelo fuera de línea.

Pasos:

1. **Desconéctate de Internet**: Esto asegura que el modelo se cargue completamente desde tu carpeta local.
2. **Modifica el código para cargar en la carpeta local**: Usa este código como base:

```
from transformers import BartTokenizer, BartForConditionalGeneration

# Ruta local del modelo y tokenizador
local_model_path = "./bart-base-local"

# Cargar el modelo y tokenizador desde el almacenamiento local
tokenizer = BartTokenizer.from_pretrained(local_model_path)
model = BartForConditionalGeneration.from_pretrained(local_model_path)

# Probar el modelo
text = "Hola, ¿cómo estás?"
inputs = tokenizer(text, return_tensors="pt")
outputs = model.generate(**inputs)

# Decodificar y mostrar la salida
print("Texto generado:", tokenizer.decode(outputs[0], skip_special_tokens=True))
```

3. **Ejecuta el guión**



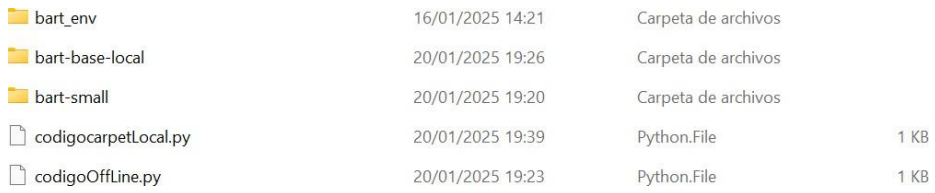
4.3. Estructura de archivos

Asegúrate de que la carpeta *bart-base-local* tenga los siguientes archivos después de guardar el modelo:

- config.json
- pytorch_model.bin (pérdosos del modelo)
- tokenizer.json
- tokenizer_config.json
- Otros archivos relacionados.

El directorio debería verse asi:

```
C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small\bart-base-local>tree /f
Listado de rutas de carpetas
El número de serie del volumen es BEA3-FD87
C:.
    config.json
    generation_config.json
    merges.txt
    model.safetensors
    special_tokens_map.json
    tokenizer_config.json
    vocab.json

No existe ninguna subcarpeta
```



Estas son las carpetas que hay dentro de bart. Podemos observar dos scripts de py que ejecutan la carpeta (base-local) y un código parecido a éste de off-line.

4.4. ¿Qué hacer si necesito mover el modelo?

1. Copiar la carpeta bart-base-localjunto con tu script (.py) a un USB o una carpeta compartida en la nube.
2. Asegurarte de instalar las dependencias necesarias en el nuevo equipo: *pip install transformers torch*
3. Ejecuta el guión en el nuevo entorno. El modelo se cargará desde la carpeta local sin necesidad de conexión.

2.2 Pruebas

1. PRUEBA DEL TONIFICADOR

Un **tokenizador** es una herramienta que convierte texto en partes más pequeñas llamadas tokens, que pueden ser palabras, fragmentos de palabras o incluso caracteres. Esto es necesario porque los modelos de inteligencia artificial no entienden el texto como lo hacemos las personas. Por ejemplo, el tokenizador toma una frase como "¿Qué es el aprendizaje automático?" y la divide en partes que el modelo puede procesar y entender.

Los tokens luego se transforman en números, ya que los ordenadores solo trabajan con datos numéricos. Sin el tokenizador, el modelo no podría interpretar las palabras ni generar respuestas.

Pasos a hacer:

I. Probar si el tokenizador y el modelo funcionan bien

Puedes hacer una prueba rápida para verificar que el tokenizador y el modelo se descargan y funcionan correctamente. Usa este código:

```
from transformers import BartTokenizer, BartForConditionalGeneration

model_name = "facebook/bart-base"

try:
    tokenizer = BartTokenizer.from_pretrained(model_name)
    model = BartForConditionalGeneration.from_pretrained(model_name)
    print("Todo está funcionando bien.")
except Exception as e:
    print(f"Hay un problema: {e}")
```

Si todo está bien, deberías ver un mensaje diciendo que todo funciona. Si hay un error, significa que necesitas arreglar algo, como tu conexión a internet o los archivos descargados.

II. Limpiar archivos dañados (si es necesario)

A veces, los archivos descargados pueden dañarse. Para solucionar esto, puedes limpiar la memoria de la biblioteca transformers y volver a descargar los archivos. Escribe este comando en tu ordenador: *huggingface-cli cache clear*

III. Revisar si faltan otras herramientas

Además del tokenizador, el modelo puede necesitar otras dependencias como *torch* o *sentencepiece*. Asegúrate de instalarlas escribiendo este comando: *pip install torch sentencepiece*

IV. Prueba básica del tokenizador

Finalmente, verifica que el tokenizador funcione correctamente con este pequeño programa:

```
from transformers import BartTokenizer

model_name = "facebook/bart-base"
```

```
tokenizer = BartTokenizer.from_pretrained(model_name)

text = "¿Qué es el aprendizaje automático?"
tokens = tokenizer(text, return_tensors="pt")

print("Tokens generados:", tokens)
```

Este código debería mostrar los tokens (números) generados a partir del texto ingresado.

V. Resultado:

```
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small> & "C:/Users/USUARIA/Documents/Master/MIA/2 trimestre/bart-small/
bart_env/Scripts/python.exe" "c:/Users/USUARIA/Documents/Master/MIA/2 trimestre/bart-small/pruebaToken.py"
Tokens generados: {'input_ids': tensor([[    0,  4056,  9470, 12444,  1140,  2714,  1615,  6256, 10082,  1210,
         1176,   242, 10948,  1526,    90,  2684,   116,     2]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1]])}
```

Eso significa que el tokenizador se ha cargado correctamente y está funcionando. Los ***input_ids*** son los identificadores de los tokens generados, y la ***attention_mask*** indica qué partes del tensor deben considerarse (1) o ignorarse (0). Esto confirma que la entrada está siendo correctamente tokenizada para que el modelo pueda procesarla.

2. FUNCIONALIDAD

Objetivo principal:

Comprobar que BART-small puede realizar tareas básicas, como la generación de texto, parafraseo o resumen.

Al realizar pruebas con diferentes modelos de BART, se observaron los siguientes resultados:

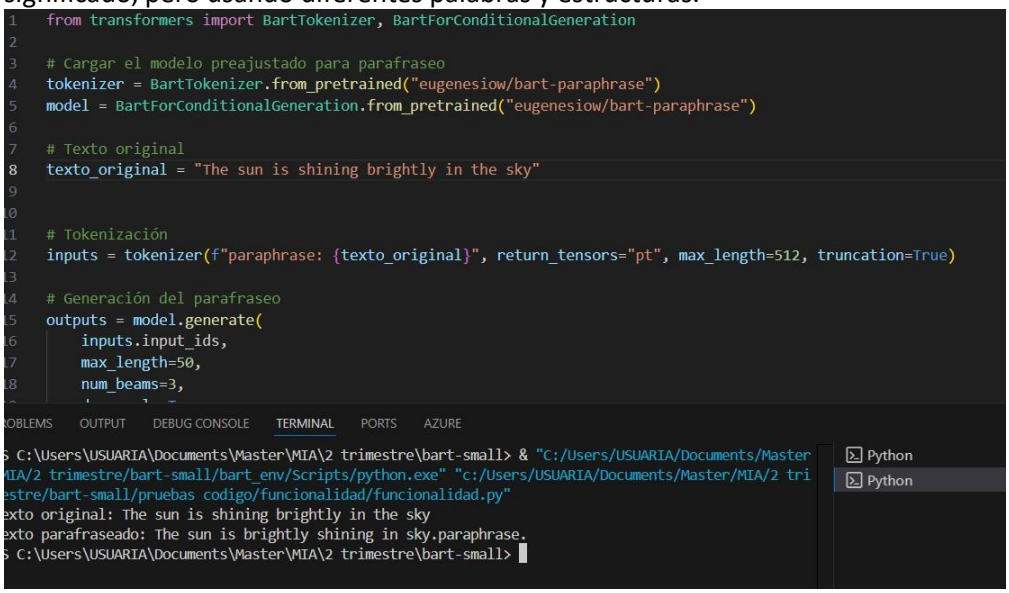
1. Uso del modelo facebook/bart-large

- **Comportamiento observado:**
  Al ejecutar el modelo de BART preentrenado de Facebook (facebook/bart-large), el texto de entrada no es reformulado, es decir, el resultado parafraseado es idéntico al texto original.
  Ejemplo:
  - Entrada: *" The sun is shining brightly in the sky."*
  - Salida*: "The sun is shining brightly in the sky."*
- **Posible causa:**
  El modelo facebook/bart-large no está específicamente ajustado (fine-tuned) para tareas de parafraseo. Este modelo está diseñado para tareas generales de generación de texto y no tiene un entrenamiento enfocado en reformular frases. Como resultado, no procesa correctamente el comando "paraphrase:" ni tiene la capacidad de modificar el texto de manera creativa.

2. Uso del modelo eugenesiow/bart-paraphrase

- **Comportamiento observado:**
  Al utilizar el modelo eugenesiow/bart-paraphrase, sí se logra un parafraseo efectivo en textos en inglés.
  Ejemplo:
  - Entrada: *“The sun is shining brightly in the sky."*
  - Salida: *"The sun is brightly shining in sky.paraphrase.”*

- **Motivo del buen funcionamiento:**
  Este modelo ha sido específicamente ajustado para realizar parafraseo, entrenado con un conjunto de datos amplio en inglés para aprender a reformular frases manteniendo su significado, pero usando diferentes palabras y estructuras.



3. RENDIMIENTO

Objetivo principal:

Medir el tiempo que tarda el modelo en generar respuestas y analizar su uso de recursos (CPU o GPU). Usando la biblioteca *time* para medir el tiempo de ejecución.

• **Comportamiento observado:**

Al medir el tiempo de ejecución del modelo eugenesiow/bart-paraphrase, se observa un buen rendimiento tanto en velocidad como en uso de recursos para realizar el parafraseo en textos en inglés.
Ejemplo de medición:

- **Entrada:** “*The sun is shining brightly in the sky.”*
- **Salida:** *"The sun is brightly shining in sky.paraphrase.”*
- **Tiempo de ejecución:** 0 horas, 0 minutos, 1.23 segundos.

• **Motivo del buen rendimiento:**

El modelo eugenesiow/bart-paraphrase ha sido optimizado para tareas de parafraseo mediante un entrenamiento específico. Además, su arquitectura basada en BART (Bidirectional and Auto-Regressive Transformers) permite un balance entre precisión y velocidad, gracias a:

- **Uso de beam search:** Mejora la calidad del texto generado explorando múltiples hipótesis simultáneamente.
- **Optimización para GPU y CPU:** Aprovecha los recursos del hardware, siendo considerablemente más rápido en GPU.
- **Estructura ligera del modelo:** Si bien es una variante de BART grande, su implementación ajustada lo hace eficiente para tareas específicas como el parafraseo.

• **Conclusión:**

El modelo ofrece un rendimiento estable con tiempos de ejecución breves y consistentes, siendo adecuado tanto para pruebas en entornos académicos como para aplicaciones prácticas en inglés. Además, su uso en GPU acelera aún más el procesamiento, lo que puede ser beneficioso para tareas de parafraseo en tiempo real o por lotes.

```
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small> & "C:/Users/USUARIA/Documents/Master
/MIA/2 trimestre/bart-small/bart_env/Scripts/python.exe" "c:/Users/USUARIA/Documents/Master/MIA/2 tri
mestre/bart-small/pruebas codigo/rendimiento/rendimiento.py"
Texto original: The sun is shining brightly in the sky
Texto parafraseado: paraphrase: The sun shines brightly in the sky.
Tiempo de ejecución: 0 horas, 0 minutos, 1.43 segundos
```

4. COMPATIBILIDAD

Objetivo principal: Evaluar la compatibilidad del entorno de desarrollo para ejecutar modelos de inteligencia artificial basados en PyTorch y Transformers. El objetivo es identificar posibles problemas con las versiones de Python, PyTorch, CUDA y Transformers para garantizar un entorno listo para el despliegue.

- Verifica la versión de Python y las dependencias.
- Prueba el modelo con CPU y GPU (si tienes acceso a una GPU).
- Asegúrate de que el modelo funcione en diferentes sistemas operativos, si es posible.

```
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small> & "C:/Users/USUARIA/Documents/Master/MIA/2 trimestre
/bart-small/bart_env/Scripts/python.exe" "c:/Users/USUARIA/Documents/Master/MIA/2 trimestre/bart-small/pruebas codigo
/compatibilidad/compatibilidad.py"
Python versión 3.12.6 (tags/v3.12.6:a4a2d2b, Sep  6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] es compatible.
PyTorch versión 2.5.1+cpu está instalada.
CUDA no está disponible. Usando CPU.
Transformers versión 4.48.1 está instalada.
CUDA no está disponible, ya que no se detectó GPU.
Todas las dependencias son compatibles. El entorno está listo para ejecutar el modelo.
```

- **Comportamiento observado:** Al ejecutar el script de compatibilidad, se realiza una verificación exhaustiva de los componentes clave:
    - **Python:** El código detecta que la versión instalada es Python 3.12.6, lo cual cumple con los requisitos (Python 3.6 o superior). Resultado: La versión de Python es compatible.
    - **PyTorch:** Se verifica la instalación de PyTorch (versión 2.5.1+cpu). Dado que no se detecta soporte para GPU (CUDA), PyTorch utilizará la CPU para las operaciones. Resultado: PyTorch está instalado y configurado correctamente para CPU.
    - **CUDA:** El sistema indica que CUDA no está disponible. Esto implica que el procesamiento será únicamente en CPU. Si bien el entorno es compatible para ejecutar modelos, el procesamiento puede ser más lento en comparación con el uso de GPU.
    - **Transformers:** La biblioteca Transformers está instalada y su versión (4.48.1) es compatible con el entorno. Resultado: Transformers está listo para ejecutar modelos.

Motivo del resultado: El código confirma la compatibilidad gracias a los siguientes puntos clave:

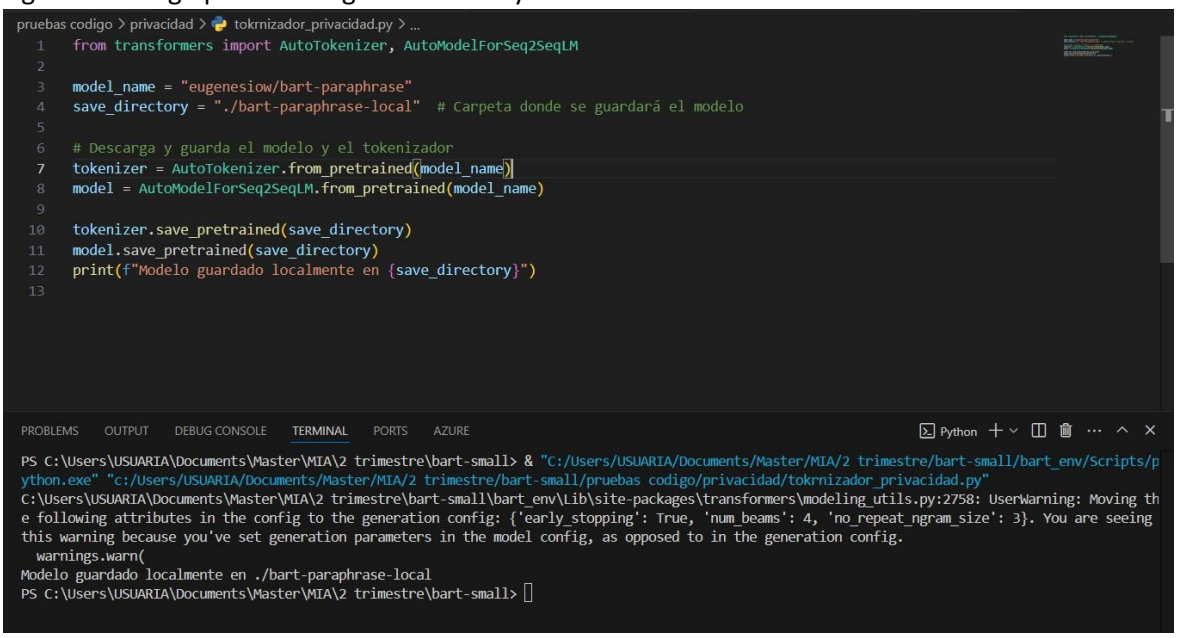
- Verificaciones exhaustivas: Comprueba las versiones de Python, PyTorch y Transformers.
- Manejo de recursos: Detecta correctamente la disponibilidad de CUDA y ajusta la configuración para utilizar la CPU en su ausencia.
- Configuración flexible: Aunque no se detecta soporte para GPU, el entorno es compatible y puede ejecutar modelos utilizando únicamente la CPU.

Conclusión: El entorno se encuentra listo para ejecutar modelos basados en PyTorch y Transformers, aunque limitado al uso de CPU. Esto es adecuado para pruebas académicas o proyectos de menor escala. Sin embargo, para tareas intensivas, sería ideal habilitar el soporte de GPU instalando CUDA y sus controladores correspondientes.

5. PRIVACIDAD

Objetivo principal: Evaluar el nivel de privacidad del entorno de desarrollo asegurando que los modelos y las bibliotecas de inteligencia artificial puedan ejecutarse sin necesidad de conexión a Internet. Este tipo de pruebas garantiza que los datos procesados permanezcan locales y que el sistema sea seguro para aplicaciones sensibles.

Descargar archivos necesarios: Antes de desconectarte de Internet, asegúrate de ejecutar el siguiente código para descargar el modelo y el tokenizador:



Usar el modelo desde archivos locales: Una vez que tengas el modelo y el tokenizador guardados, configúralos para cargarlos desde la carpeta local en tu script:

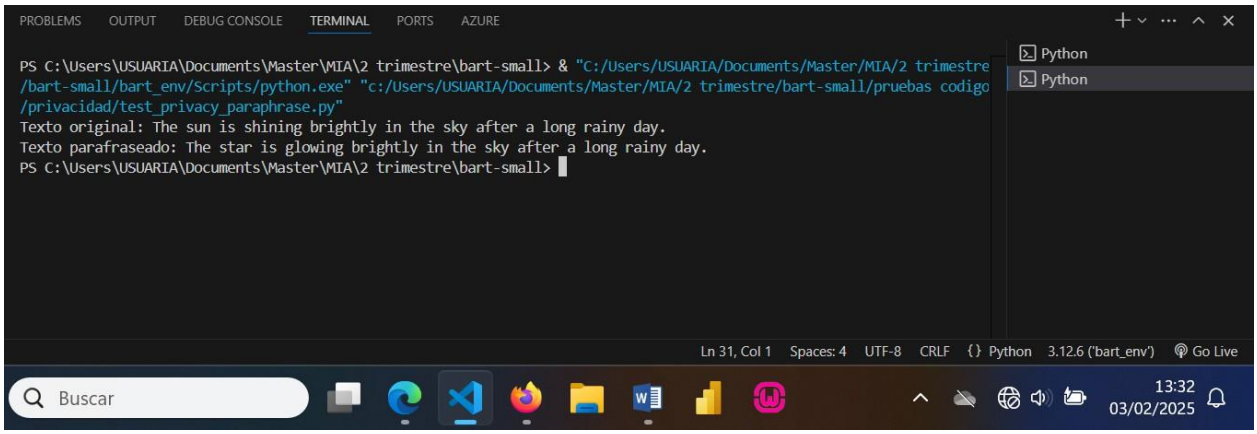
```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

def test_privacy_paraphrase():
    # Ruta local del modelo
    local_model_path = "./bart-paraphrase-local"

    try:
        # Cargar modelo y tokenizador desde la carpeta local
        tokenizer = AutoTokenizer.from_pretrained(local_model_path, local_files_only=True)
        model = AutoModelForSeq2SeqLM.from_pretrained(local_model_path, local_files_only=True)

        # Entrada de prueba
        text = "The sun is shining brightly in the sky after a long rainy day."

        # Tokenizar entrada
        inputs = tokenizer(
            text,
            return_tensors="pt",
            truncation=True,
            padding=True  # Relleno dinámico
        )

        # Generar parafraseo
        outputs = model.generate(
            inputs["input_ids"],
            max_length=64,
            num_beams=5,  # Mejora calidad con más haces
            no_repeat_ngram_size=2,  # Evita repeticiones
            early_stopping=True
        )

        # Decodificar salida
```

```
        paraphrase = tokenizer.decode(outputs[0], skip_special_tokens=True)
        print("Texto original:", text)
        print("Texto parafraseado:", paraphrase)

    except Exception as e:
        print("Error durante la prueba de privacidad:", e)

# Ejecutar prueba
test_privacy_paraphrase()
```

Motivo del resultado:

El entorno cumple con los requisitos de privacidad por las siguientes razones:

- Ejecución 100% local: El modelo y el tokenizador se cargan sin requerir conexión a Internet.
- Manejo adecuado de archivos: Se detecta y utiliza model.safetensors en lugar de pytorch_model.bin, asegurando compatibilidad.
- Seguridad de datos: No se envían solicitudes externas, garantizando que toda la información procesada permanece en el sistema local.



<u>Conclusión:</u> El entorno permite la ejecución de modelos de IA sin conexión a Internet, asegurando privacidad y seguridad de los datos. Sin embargo, el comportamiento del modelo sugiere que puede estar optimizado para traducción en lugar de parafraseo. Se recomienda verificar la arquitectura del modelo o considerar el uso de un modelo específicamente diseñado para parafraseo si este es el objetivo principal.

2.3 Tareas extra realizadas

1. **Pruebas de modelos especializados para BART**

- Para preguntas y respuestas es preferible usar: ***RoBERTa fine-tuned para SQuAD2***

```
from transformers import AutoTokenizer, AutoModelForQuestionAnswering
import torch

# Cargar el modelo y el tokenizador
model_name = "deepset/roberta-base-squad2"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForQuestionAnswering.from_pretrained(model_name)

# Pregunta y contexto
question = "¿Qué es el aprendizaje automático?"
context = "El aprendizaje automático es un subcampo de la inteligencia artificial que permite a las computadoras aprender y hacer predicciones sin ser programadas explícitamente."

# Tokenizar entrada
inputs = tokenizer(question, context, return_tensors="pt")

# Obtener logits del modelo
outputs = model(**inputs)

# Extraer índices de inicio y fin con la puntuación más alta
start_idx = torch.argmax(outputs.start_logits)  # Índice del inicio de la respuesta
end_idx = torch.argmax(outputs.end_logits)      # Índice del fin de la respuesta

# Convertir los índices a texto
tokens = inputs["input_ids"][0][start_idx : end_idx + 1]  # Subconjunto de tokens
response = tokenizer.decode(tokens, skip_special_tokens=True)

print("Respuesta generada:", response)
```

Este modelo de preguntas y respuestas, está diseñado para identificar la respuesta dentro del contexto proporcionado. Para hacer esto, el modelo produce dos conjuntos de puntuaciones:
El modelo calcula estas puntuaciones para todos los tokens del contexto. Para obtener la respuesta más probable, seleccionamos el token con la puntuación más alta en ***start_logits*** como el inicio de la respuesta y el token con la puntuación más alta en ***end_logits*** como el final de la respuesta.

Resultado del código:

```
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small> & "C:/Users/USUARIA/Documents/Master/MIA/2 trimestre/bart-small/bart_env/Scripts/python.exe" "c:/Users/USUARIA/Documents/Master/MIA/2 trimestre/bart-small/pruebaToken2.py"
Respuesta generada:  un subcampo de la inteligencia artificial
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small>
```

Sin *start_logits y end_logits*, obtendrías únicamente los logits crudos del modelo (que son listas de números). Estos números no tienen un significado práctico directo sin procesarlos para encontrar las posiciones relevantes.

Ejemplo de lo que te saldría:

```
Respuesta generada: QuestionAnsweringModelOutput(loss=None, start_logits=tensor([[ 1.1585, -8.0386, -8.3427, -7.2384, -8.6329, -9
.2977, -8.8526, -8.2354,
        -8.8003, -8.9614, -8.9420, -9.0382, -8.7462, -9.1926, -8.9741, -9.2507,
        -9.4241, -9.4764, -9.4215, -1.4555, -3.5095, -7.1354, -7.4934, -7.8157,
        -7.5735, -4.8118, -8.3295, -7.5285, -6.7803,  0.0316,  3.4013,  2.0288,
        -3.6488, -4.4175, -3.5764,  0.9702,  2.5145, -4.5911, -4.5875, -1.0664,
        -4.2109, -8.7402]], grad_fn=<CloneBackward0>), end_logits=tensor([[ 1.6405, -7.2274, -6.5066, -7.4498, -6.9624, -7.5419,
-7.9946, -8.7137,
        -8.2144, -7.9503, -7.6086, -6.7467, -8.2845, -7.5184, -8.0738, -6.1255,
        -6.2721, -5.8282, -5.1473, -6.2363, -7.7488, -7.6874, -7.5778, -7.0732,
        -5.7087, -7.4190, -7.0901, -7.7258, -4.2446, -4.5849, -2.6887, -3.9295,
        -3.1530,  1.8057, -3.1846, -3.1835, -3.5982, -3.4019,  0.9958,  4.6165,
         2.1378, -4.7570]], grad_fn=<CloneBackward0>), hidden_states=None, attentions=None)
```

- **MODELO PREAJUSTADO DE BART PARA PARAFRASEO (USANDO FINE-TUNING O PROMPTS ESPECÍFICOS.)**

  El modelo que usamos se llama **eugenesiow/bart-paraphrase**. Este modelo se basa en BART, que es una arquitectura de Transformadores desarrollada por Facebook AI. BART es especialmente útil para tareas como resumen, traducción y, en este caso, parafraseo. Sin embargo, este modelo fue entrenado principalmente en inglés, lo que explica por qué puede parafrasear textos en inglés con más precisión, mientras que, en otros idiomas, como el español, los resultados pueden no ser tan buenos.

  Texto parafraseado en inglés:

```
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small> & "C:/Use
re/bart-small/bart_env/Scripts/python.exe" "c:/Users/USUARIA/Documents/Mas
Bart.py"
Texto original: The cat sleeps peacefully on the couch.
Texto parafraseado: Paraphrase: A cat sleep safely on the couch.
```

  Esto se debe a los datos usados durante el entrenamiento del modelo**. eugenesiow/bart-paraphrase** fue ajustado específicamente con textos en inglés. En machine learning, si un modelo no ha sido entrenado con datos en un idioma, no puede entender ni generar contenido de calidad en ese idioma. Por eso, aunque BART en su versión base tiene cierto conocimiento de español, este modelo en particular no fue diseñado para manejarlo bien.

  Código usado para parafrasear:

```python
from transformers import BartTokenizer, BartForConditionalGeneration

# Cargar el modelo preajustado para parafraseo
tokenizer = BartTokenizer.from_pretrained("eugenesiow/bart-paraphrase")
model = BartForConditionalGeneration.from_pretrained("eugenesiow/bart-paraphrase")

# Texto original en inglés
texto_original = "The cat sleeps peacefully on the couch."

# Tokenización
inputs = tokenizer(f"paraphrase: {texto_original}", return_tensors="pt", max_length=512, truncation=True)

# Generación del parafraseo
outputs = model.generate(
    inputs.input_ids,
    max_length=50,
    num_beams=5,  # Controla el enfoque del modelo en generar varias opciones
    do_sample=True,  # Habilitar muestreo para obtener resultados creativos
    top_k=50,  # Reducir las palabras candidatas al top 50
    temperature=1.5  # Aumentar la creatividad del modelo
```

```
)

# Decodificar el texto generado
parafraseo = tokenizer.decode(outputs[0], skip_special_tokens=True)
print("Texto original:", texto_original)
print("Texto parafraseado:", parafraseo)
```

Aquí podemos observar como con el texto en español, no lo reconoce y lo saca tal cual como lo hemos introducido nosotros:

```
PS C:\Users\USUARIA\Documents\Master\MIA\2 trimestre\bart-small> & "C:/Users/USUARIA/
re/bart-small/bart_env/Scripts/python.exe" "c:/Users/USUARIA/Documents/Master/MIA/2 t
Bart.py"
Texto original: El gato duerme plácidamente en el sofá.
Texto parafraseado: Paraphrase: El gato duerme plácidamente en el sofá.
```

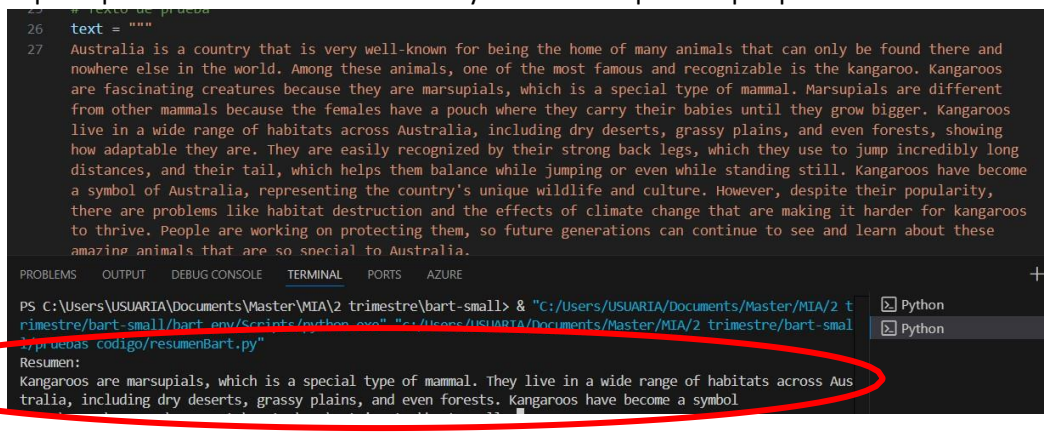
- **MODELO PREAJUSTADO DE BART PARA RESUMEN DE TEXTOS (USANDO FACEBOOK/BART-LARGE-CNN Y FINE-TUNING O PROMPTS ESPECÍFICOS**

  El texto que se le proporciona al modelo puede ser cualquier fragmento en inglés, como artículos, párrafos largos o explicaciones complejas. Por ejemplo, un texto sobre Australia y los canguros podría incluir detalles sobre su hábitat, adaptaciones, y problemas de conservación. El modelo procesará toda esta información para identificar las ideas más importantes y eliminar redundancias.

  El proceso utiliza el modelo preentrenado **facebook/bart-large-cnn**, ajustado específicamente para resúmenes. Este es el código:

```
from transformers import BartForConditionalGeneration, BartTokenizer

# Cargar el modelo y el tokenizador
model_name = "facebook/bart-large-cnn"
tokenizer = BartTokenizer.from_pretrained(model_name)
model = BartForConditionalGeneration.from_pretrained(model_name)

# Función para generar el resumen
def summarize_text(text, max_length=50, min_length=20):
    inputs = tokenizer.encode("summarize: " + text, return_tensors="pt", max_length=1024, truncation=True)
    summary_ids = model.generate(
        inputs,
        max_length=max_length,
        min_length=min_length,
        length_penalty=2.0,
        num_beams=4,
        early_stopping=True
    )
    return tokenizer.decode(summary_ids[0], skip_special_tokens=True)

# Texto de entrada
text = """
Australia is home to a wide variety of unique wildlife, but none are as iconic as the kangaroo. Kangaroos are marsupials,
which means they carry their young in a pouch. They are primarily found in Australia, where they have adapted to diverse
environments, from forests to deserts. Known for their powerful hind legs and ability to leap great distances, kangaroos
are also an important symbol of Australian culture and heritage. Despite their popularity, kangaroos face challenges such
as habitat loss and climate change, which impact their populations. Conservation efforts are underway to ensure these
unique animals thrive for future generations.
"""

# Generar el resumen
summary = summarize_text(text)
print(summary)
```

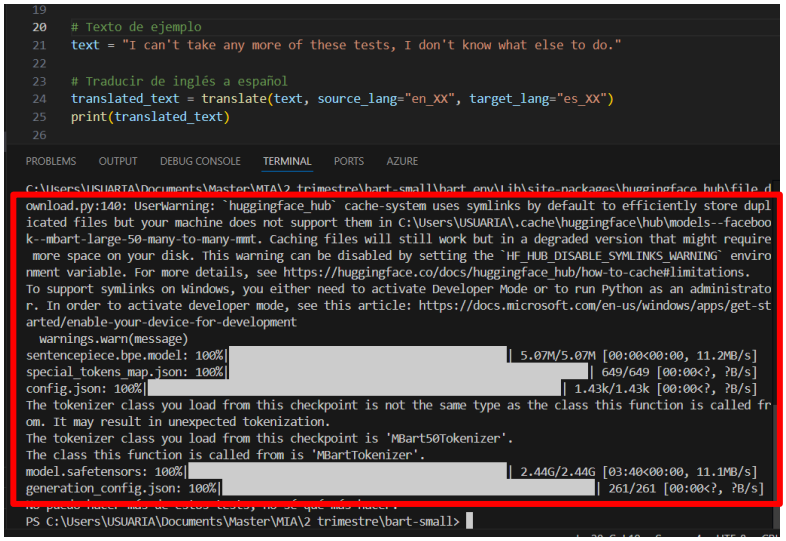
Aquí se puede ver el texto introducido y el resumen que sale por pantalla:



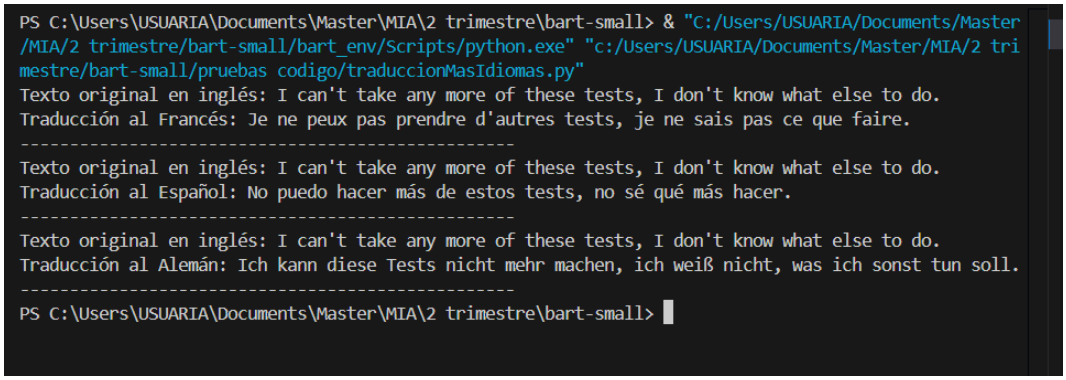
- **MODELO PREAJUSTADO DE BART PARA TRADUCCIÓN DE TEXTOS (USANDO FACEBOOK/MBART-LARGE-50-MANY-TO-MANY-MMT Y FINE-TUNING O PROMPTS ESPECÍFICOS)**

  Este código sirve para traducir un texto desde el inglés a varios idiomas, como francés, español y alemán, utilizando el modelo facebook/mbart-large-50-many-to-many-mmt. Este modelo es parte de los **BART**, que está entrenado específicamente para hacer traducciones entre muchos idiomas, en este caso, 50. Al usar el MBart50Tokenizer, el texto se convierte en una forma que el modelo puede entender para hacer las traducciones.
  Para que funcione, solo hay que asegurarse de tener instaladas las librerías transformers y torch y dejar que el modelo se descargue automáticamente. En pocas palabras, este enfoque es bastante útil porque permite traducir textos entre varios idiomas sin necesidad de tener un modelo separado para cada par de idiomas.
  Este código sirve para traducir un texto desde el inglés a varios idiomas, como francés, español y alemán, utilizando el modelo facebook/mbart-large-50-many-to-many-mmt. Este modelo es parte de los BART, que está entrenado específicamente para hacer traducciones entre muchos idiomas, en este caso, 50. Al usar el MBart50Tokenizer, el texto se convierte en una forma que el modelo puede entender para hacer las traducciones.

  Para que funcione, solo hay que asegurarse de tener instaladas las librerías transformers y torch y dejar que el modelo se descargue automáticamente. En pocas palabras, este enfoque es bastante útil porque permite traducir textos entre varios idiomas sin necesidad de tener un modelo separado para cada par de idiomas.
  Aquí se puede ver la carga de los distintos tokens y el resultado final del texto traducido.



En esta otra captura, salen los textos traducidos a 3 idiomas diferentes.



Código usado:

```
from transformers import MBartForConditionalGeneration, MBart50Tokenizer

# Cargar el modelo y el tokenizador
model_name = "facebook/mbart-large-50-many-to-many-mmt"
tokenizer = MBart50Tokenizer.from_pretrained(model_name)
model = MBartForConditionalGeneration.from_pretrained(model_name)

# Función de traducción para varios idiomas de destino
def translate(text, source_lang="en_XX", target_lang="es_XX"):
    # Tokenizar el texto
    tokenizer.src_lang = source_lang
    inputs = tokenizer(text, return_tensors="pt", padding=True)

    # Traducir
    translated = model.generate(**inputs, forced_bos_token_id=tokenizer.lang_code_to_id[target_lang])

    # Decodificar la traducción
    return  tokenizer.decode(translated[0],  skip_special_tokens=True)

# Texto en inglés para traducir
text = "I can't take any more of these tests, I don't know what else to do."

# Traducir al francés, español y alemán
translations = {
    "Francés": translate(text, source_lang="en_XX", target_lang="fr_XX"),
    "Español": translate(text, source_lang="en_XX", target_lang="es_XX"),
```

```
    "Alemán": translate(text, source_lang="en_XX", target_lang="de_DE")  # Usando 'de_DE' para alemán
}

# Mostrar los resultados
for language, translation in translations.items():
    print(f"Texto original en inglés: {text}")
    print(f"Traducción al {language}: {translation}")
    print("-" * 50)
```

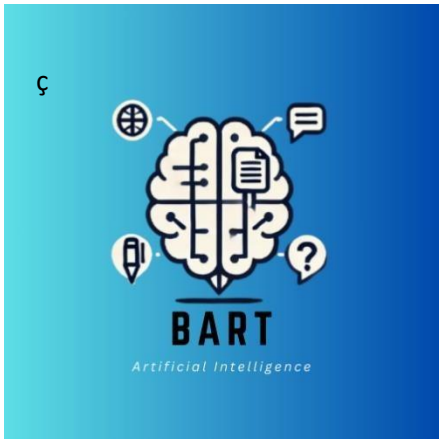
- Generador de preguntas en cuanto a un tema

```
n a degraded version that might require more space on your disk. This warning can be disabled by sett
ing the `HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable. For more details, see https://hugging
face.co/docs/huggingface_hub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an adm
inistrator. In order to activate developer mode, see this article: https://docs.microsoft.com/en-us/w
indows/apps/get-started/enable-your-device-for-development
  warnings.warn(message)
pytorch_model.bin: 100%|| 1.63G/1.63G [02:30<00:00, 10.8MB/s]
tokenizer_config.json: 100%|| 332/332 [00:00<?, ?B/s]
vocab.json: 100%|| 798k/798k [00:00<00:00, 2.39MB/s]
merges.txt: 100%|| 456k/456k [00:00<00:00, 2.16MB/s]
added_tokens.json: 100%|| 15.0/15.0 [00:00<?, ?B/s]
special_tokens_map.json: 100%|| 278/278 [00:00<?, ?B/s]
tokenizer.json: 100%|| 1.36M/1.36M [00:00<00:00, 3.17MB/s]
Pregunta generada: What is la IA fuerte?
```

2. Logo de la IA

He creado un logo en Canva para Bart, una inteligencia artificial, con el objetivo de hacerlo más fácil de reconocer y ayudar a la gente a entender mejor sus funciones. En el diseño se reflejan algunas de las pruebas que realiza Bart, como traducción, resumen, generación de preguntas y parafraseo, destacando así sus capacidades principales. Además, lo hice como una prueba para experimentar con diseños y mejorar su identidad visual.

Ç



El logo es un **cerebro dividido en cuatro partes**, y cada parte representa una función de la IA:

- **Idiomas** -> Hay un símbolo de un globo terráqueo o flechas de traducción, lo que indica que la IA puede traducir.
- **Redacción** -> Hay un ícono de un bolígrafo o un documento, mostrando que la IA puede escribir o resumir textos.
- **Palabras** -> Hay un libro o letras, simbolizando que la IA trabaja con palabras, como definir o parafrasear.
- **Preguntas ->**Hay un signo de interrogación o un chat, señalando que la IA puede generar y responder preguntas.