

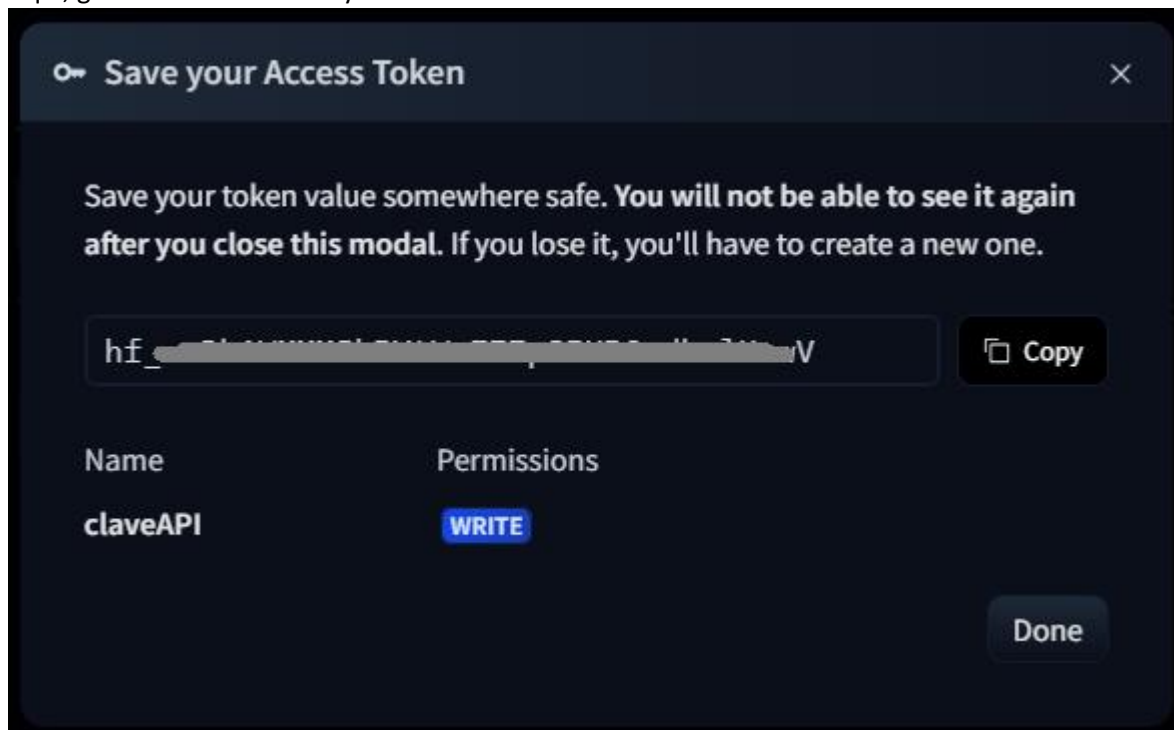
# Ejercicios Capítulo 7

1. Escoger dos de los siguientes ejercicios para su realización. Los ejercicios desarrollados deben ir muy bien documentados.

1. **Generación de Texto con GPT-3:** Realizar ejercicios de generación de texto utilizando una API basada en GPT-3. Pueden generar historias, poemas o respuestas a preguntas específicas.

Para empezar, buscamos la conexión a una API de GPT-3.

Para poder establecer una generación de texto mediante API, usaremos Hugging Face y su API. Aquí, generamos una API key:



Esta se usará más adelante para tener acceso correcto a los recursos de la API

Se instalan las dependencias necesarias mediante los siguientes comandos:

## pip install transformers

```
C:\Users\Microsoft Windows>pip install transformers
Collecting transformers
  Downloading transformers-4.49.0-py3-none-any.whl.metadata (44 kB)
Collecting filelock (from transformers)
  Downloading filelock-3.17.0-py3-none-any.whl.metadata (2.9 kB)
Collecting huggingface-hub<1.0,>=0.26.0 (from transformers)
  Downloading huggingface-hub-0.29.2-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: numpy<=1.17 in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging<=20.0 in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from transformers) (24.1)
Collecting pyyaml<=5.1 (from transformers)
  Downloading PyYAML-6.0.2-cp312-cp312-win amd64.whl.metadata (2.1 kB)
Collecting regex<=2019.12.17 (from transformers)
  Downloading regex-2024.11.6-cp312-cp312-win amd64.whl.metadata (41 kB)
Collecting requests (from transformers)
  Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting tokenizers<0.22,>=0.21 (from transformers)
  Downloading tokenizers-0.21.0-cp39-abi3-win amd64.whl.metadata (6.9 kB)
Collecting safetensors<0.4.1 (from transformers)
  Downloading safetensors-0.5.3-cp39-abi3-win amd64.whl.metadata (3.9 kB)
Requirement already satisfied: tqdm<=4.27 in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from transformers) (4.67.1)
Collecting fsspec<=2023.5.0 (from huggingface-hub<1.0,>=0.26.0->transformers)
  Downloading fsspec-2023.2.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from huggingface-hub<1.0,>=0.26.0->transformers) (4.12.2)
Requirement already satisfied: colorama in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from tqdm<=4.27->transformers) (0.4.6)
Collecting charset-normalizer<4,>=2 (from requests->transformers)
  Downloading charset-normalizer-3.4.1-cp312-cp312-win amd64.whl.metadata (36 kB)
Requirement already satisfied: idna<4,>=2.5 in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from requests->transformers) (3.10.1)
Collecting urllib3<3,>=1.21.1 (from requests->transformers)
  Downloading urllib3-2.3.0-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from requests->transformers) (2025.1.31)
Downloading transformers-4.49.0-py3-none-any.whl (10.0 MB)
10.0/10.0 MB 14.1 MB/s eta 0:00:00
Downloading huggingface-hub-0.29.2-py3-none-any.whl (468 kB)
Downloading PyYAML-6.0.2-cp312-cp312-win amd64.whl (156 kB)
Downloading regex-2024.11.6-cp312-cp312-win amd64.whl (273 kB)
Downloading safetensors-0.5.3-cp39-abi3-win amd64.whl (308 kB)
Downloading tokenizers-0.21.0-cp39-abi3-win amd64.whl (2.4 MB)
2.4/2.4 MB 11.3 MB/s eta 0:00:00
Downloading filelock-3.17.0-py3-none-any.whl (16 kB)
Downloading requests-2.32.3-py3-none-any.whl (64 kB)
Downloading charset-normalizer-3.4.1-cp312-cp312-win amd64.whl (102 kB)
Downloading fsspec-2023.2.0-py3-none-any.whl (184 kB)
Downloading urllib3-2.3.0-py3-none-any.whl (128 kB)
Installing collected packages: urllib3, safetensors, regex, pyyaml, fsspec, filelock, charset-normalizer, requests, huggingface-hub, tokenizers, transformers
Successfully installed charset-normalizer-3.4.1 filelock-3.17.0 fsspec-2023.2.0 huggingface-hub-0.29.2 pyyaml-6.0.2 regex-2024.11.6 requests-2.32.3 safetensors-0.5.3 tokenizers-0.21.0 transformers-4.49.0 urllib3-2.3.0
```

## pip install torch torchvision torchaudio

```
C:\Users\Microsoft Windows>pip install torch torchvision torchaudio
Collecting torch
  Downloading torch-2.6.0-cp312-cp312-win amd64.whl.metadata (28 kB)
Collecting torchvision
  Downloading torchvision-0.21.0-cp312-cp312-win amd64.whl.metadata (6.3 kB)
Collecting torchaudio
  Downloading torchaudio-2.6.0-cp312-cp312-win amd64.whl.metadata (6.7 kB)
Requirement already satisfied: filelock in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from torch) (3.17.0)
Requirement already satisfied: typing-extensions>=4.10.0 in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from torch) (4.12.2)
Collecting networkx (from torch)
  Downloading networkx-3.4.2-py3-none-any.whl.metadata (6.3 kB)
Collecting jinja2 (from torch)
  Downloading jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Requirement already satisfied: fsspec in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from torch) (2025.2.0)
Requirement already satisfied: setuptools in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from torch) (69.5.1)
Collecting sympy==1.13.1 (from torch)
  Downloading sympy-1.13.1-py3-none-any.whl.metadata (12 kB)
Collecting mpmath<1.4,>=1.1.0 (from sympy==1.13.1->torch)
  Downloading mpmath-1.3.0-py3-none-any.whl.metadata (8.6 kB)
Requirement already satisfied: numpy in c:\users\microsoft windows\appdata\local\programs\python\python312\lib\site-packages (from torchvision) (1.26.4)
Collecting pillow<8.3.4,>=5.3.0 (from torchvision)
  Downloading pillow-11.1.0-cp312-cp312-win amd64.whl.metadata (9.3 kB)
Collecting MarkupSafe>=2.0 (from jinja2->torch)
  Downloading MarkupSafe-3.0.2-cp312-cp312-win amd64.whl.metadata (4.1 kB)
Downloading torch-2.6.0-cp312-cp312-win amd64.whl (204.1 MB)
204.1/204.1 MB 8.9 MB/s eta 0:00:00
Downloading sympy-1.13.1-py3-none-any.whl (6.2 MB)
6.2/6.2 MB 9.7 MB/s eta 0:00:00
Downloading torchvision-0.21.0-cp312-cp312-win amd64.whl (1.6 MB)
1.6/1.6 MB 9.2 MB/s eta 0:00:00
Downloading torchaudio-2.6.0-cp312-cp312-win amd64.whl (2.4 MB)
2.4/2.4 MB 11.7 MB/s eta 0:00:00
Downloading pillow-11.1.0-cp312-cp312-win amd64.whl (2.6 MB)
2.6/2.6 MB 10.8 MB/s eta 0:00:00
Downloading jinja2-3.1.6-py3-none-any.whl (134 kB)
Downloading networkx-3.4.2-py3-none-any.whl (1.7 MB)
1.7/1.7 MB 11.7 MB/s eta 0:00:00
Downloading MarkupSafe-3.0.2-cp312-cp312-win amd64.whl (15 kB)
Downloading mpmath-1.3.0-py3-none-any.whl (536 kB)
536.2/536.2 kB 5.8 MB/s eta 0:00:00
Installing collected packages: mpmath, sympy, pillow, networkx, MarkupSafe, jinja2, torch, torchvision, torchaudio
Successfully installed MarkupSafe-3.0.2 jinja2-3.1.6 mpmath-1.3.0 networkx-3.4.2 pillow-11.1.0 sympy-1.13.1 torch-2.6.0 torchaudio-2.6.0 torchvision-0.21.0

[notice] A new release of pip is available: 24.2 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Hecho esto, podemos generar un código que, usando la librería Transformers, se realice una correcta generación de texto:

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer
import torch

# Configurar tu clave API
access_token = 'tu_clave_api_aqui'

#Cargar el modelo y el tokenizador
model_name = "gpt2"

model = GPT2LMHeadModel.from_pretrained(model_name, token=access_token)
tokenizer = GPT2Tokenizer.from_pretrained(model_name, token=access_token)

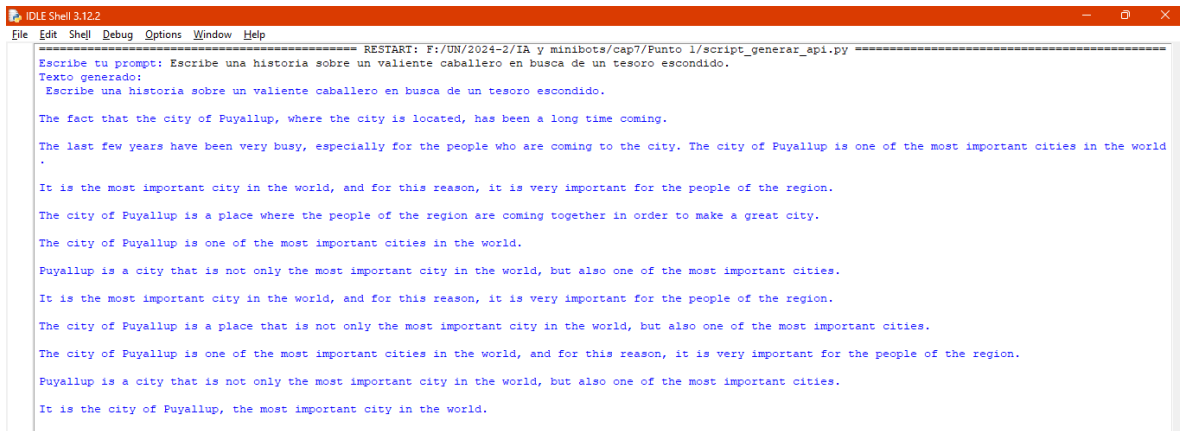
#Función para generar texto
def generar_texto(prompt, max_length=1000, temperature=0.7, top_k=50, top_p=0.9):
    inputs = tokenizer.encode(prompt, return_tensors="pt")
    attention_mask = torch.ones(inputs.shape, dtype=torch.long) # Crear la attention_mask
    outputs = model.generate(
        inputs,
        attention_mask=attention_mask,
        max_length=max_length,
        pad_token_id=tokenizer.eos_token_id,
        temperature=temperature,
        top_k=top_k,
        top_p=top_p,
        do_sample=True, #Habilitar muestreo
        eos_token_id=tokenizer.eos_token_id # Agregar el token de fin de secuencia
    )
    texto_generado = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return texto_generado

#Solicitar prompt
prompt = input("Escribe tu prompt: ")
texto_generado = generar_texto(prompt)
```

```
print("Texto generado:\n", texto_generado)
```

Cabe denotar que, en versiones gratuitas y por la complejidad del generador, solo admite hasta 1023 caracteres de salida, he aquí el ejemplo ante el prompt “Escribe una historia sobre un valiente caballero en busca de un tesoro escondido.”

Así, la salida resulta en esto:



```
===== RESTART: F:\UN\2024-2\IA y minibots\cap7\Punto 1/script_generar_api.py =====
Escribe tu prompt: Escribe una historia sobre un valiente caballero en busca de un tesoro escondido.
Texto generado:
Escribe una historia sobre un valiente caballero en busca de un tesoro escondido.

The fact that the city of Puyallup, where the city is located, has been a long time coming.

The last few years have been very busy, especially for the people who are coming to the city. The city of Puyallup is one of the most important cities in the world
.

It is the most important city in the world, and for this reason, it is very important for the people of the region.

The city of Puyallup is a place where the people of the region are coming together in order to make a great city.

The city of Puyallup is one of the most important cities in the world.

Puyallup is a city that is not only the most important city in the world, but also one of the most important cities.

It is the most important city in the world, and for this reason, it is very important for the people of the region.

The city of Puyallup is a place that is not only the most important city in the world, but also one of the most important cities.

The city of Puyallup is one of the most important cities in the world, and for this reason, it is very important for the people of the region.

Puyallup is a city that is not only the most important city in the world, but also one of the most important cities.

It is the city of Puyallup, the most important city in the world.
```

Para ahondar en detalles sobre esta generación de texto:

La API de inferencia sin servidor ofrece una forma rápida y sencilla de explorar miles de modelos para una variedad de tareas. Ya sea que esté creando un prototipo de una nueva aplicación o experimentando con capacidades de aprendizaje automático, esta API brinda acceso instantáneo a modelos de alto rendimiento en múltiples dominios:

- Generación de texto: incluye modelos de lenguaje grandes y pautas para llamar a herramientas, para generar y experimentar con respuestas de alta calidad.
- Generación de imágenes: cree fácilmente imágenes personalizadas, incluidas LoRA para sus propios estilos.
- Incrustaciones de documentos: cree sistemas de búsqueda y recuperación con incrustaciones SOTA.
- Tareas de IA clásica: modelos listos para usar para clasificación de texto, clasificación de imágenes, reconocimiento de voz y más.

Beneficios clave:

- Prototipado instantáneo: acceda a modelos potentes sin necesidad de configuración.
- Diversos casos de uso: una API para texto, imágenes y más.
- Amigable para desarrolladores: solicitudes simples, respuestas rápidas.

### **API de listado de repositorios:**

Al realizar llamadas a la API para recuperar información sobre los repositorios, el `createdAt` atributo indica el momento en que se creó el repositorio respectivo. Es importante tener en cuenta que existe un valor único, `2022-03-02T23:29:04.000Z` asignado a todos los repositorios que se crearon antes de que comenzáramos a almacenar las fechas de creación.

Por otro lado, también se ejecutan endpoints que administran configuraciones del repositorio, como la creación y eliminación de un repositorio.

### **Transformers**

En cuanto a Transformers, se usó aprendizaje automático para PyTorch , TensorFlow y JAX. Proporcionan API y herramientas para descargar y entrenar fácilmente modelos preentrenados de última generación. El uso de modelos preentrenados puede reducir los costos de procesamiento y los recursos necesarios para entrenar un modelo desde cero. Estos modelos admiten tareas comunes en diferentes modalidades, como:

- Procesamiento del lenguaje natural : clasificación de texto, reconocimiento de entidades con nombre, respuesta a preguntas, modelado del lenguaje, generación de código, resumen, traducción, opción múltiple y generación de texto.
- Visión artificial : clasificación de imágenes, detección de objetos y segmentación.
- Audio : reconocimiento automático de voz y clasificación de audio.
- Multimodal : respuesta a preguntas de tabla, reconocimiento óptico de caracteres, extracción de información de documentos escaneados, clasificación de vídeo y respuesta a preguntas visuales.

Los transformadores admiten la interoperabilidad de marcos entre PyTorch, TensorFlow y JAX. Esto brinda la flexibilidad de usar un marco diferente en cada etapa de la vida de un modelo; entrenar un modelo en tres líneas de código en un marco y cargarlo para inferencia en otro. Los modelos también se pueden exportar a un formato como ONNX y TorchScript para su implementación en entornos de producción.

### **Datasets:**

Permiten acceder y compartir fácilmente conjuntos de datos para tareas de audio, visión artificial y procesamiento del lenguaje natural (PNL). Con el respaldo del formato Apache Arrow, se procesan grandes conjuntos de datos con lecturas de copia cero sin restricciones de memoria para lograr una velocidad y eficiencia óptimas. También contamos con una integración profunda con Hugging Face Hub , lo que le permite cargar y compartir fácilmente un conjunto de datos con aprendizaje automático más amplio.

# Ejercicio Creativo: Generador de cuentos Interactivos

Este proyecto buscaría dar a entender a un usuario cómo funcionan los modelos generativos basados en Transformers; facilitar el aprendizaje e interactuar con estos modelos para generar contenido creativo; y el desarrollo de habilidades en el procesamiento de lenguaje natural y la generación de texto.

Para esto, se desarrolla una aplicación que permita a los usuarios co-crear cuentos interactivos. Utiliza un modelo generativo basado en Transformers (como GPT-3 o GPT-2) para continuar las historias basadas en las entradas del usuario.

Se empieza preparando el entorno con la instalación las bibliotecas necesarias (`transformers`, `torch`, etc.) como en el anterior ejercicio. Después, se genera un cuento inicial, creando una función que genere el inicio de un cuento:

```
def generar_inicio_cuento():
```

```
    prompt = "Había una vez en un reino lejano, un valiente caballero llamado"
```

```
    inicio_cuento = generar_texto(prompt) # Usar tu función de generación de texto
```

```
    return inicio_cuento
```

Asimismo, se solicita al usuario que escriba la siguiente línea del cuento, y se usa la entrada del usuario para generar la continuación del cuento.

```
while True:
```

```
    entrada_usuario = input("Escribe la siguiente línea del cuento: ")
```

```
    prompt = cuento_actual + "\n" + entrada_usuario
```

```
    continuation = generar_texto(prompt)
```

```
    cuento_actual += "\n" + continuation
```

```
    print("Cuento hasta ahora:\n", cuento_actual)
```

Después, se puede guardar y compartir el cuento en un archivo para el usuario:

```
with open("cuento_interactivo.txt", "w") as file:
```

```
    file.write(cuento_actual)
```

```
print("Cuento guardado en 'cuento_interactivo.txt'")
```

¿Resultado? Un cuento interactivo co-creado por el usuario y la IA, que puede ser guardado y compartido.