Objetivos

- Generar respuestas con un LLM gratuito.
- Explicar con LIME qué palabras favorecen estereotipos.
- Comparar prompts con bias vs neutrales.

Celda 1: Instalación de librerías

Esta celda instala LIME (para explainability) y Transformers (para el modelo GPT2).

```
!pip install -q lime transformers
print("¡Librerías instaladas!")
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275.7/275.7 kB 7.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
  Building wheel for lime (setup.py) ... done
¡Librerías instaladas!
```

Celda 2: Cargar el modelo GPT2-small

Cargamos GPT2-small, un modelo generativo ligero (117M parámetros) que funciona en Colab gratuito. Es ideal para demos – genera texto a partir de prompts.

```
from transformers import AutoTokenizer, AutoModelForCausalLM
import torch

model_name = "gpt2"
```

```
model_name = "gpt2"

tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)
model.eval()

print("¡GPT2-small cargado!")
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/se
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 1.63kB/s]
config.json: 100%  665/665 [00:00<00:00, 16.7kB/s]
vocab.json: 100%  1.04M/1.04M [00:00<00:00, 6.29MB/s]
merges.txt: 100%  456k/456k [00:00<00:00, 4.32MB/s]
tokenizer.json: 100%  1.36M/1.36M [00:00<00:00, 18.4MB/s]
WARNING:torchao.kernel.intmm:Warning: Detected no triton, on systems without Triton certain kernels will
model.safetensors: 100%  548M/548M [00:03<00:00, 279MB/s]
generation_config.json: 100%  124/124 [00:00<00:00, 5.85kB/s]
¡GPT2-small cargado!
```

Celda 3: Función para generar respuestas

Esta función genera texto a partir del prompt, usando el modelo GPT2. Es el corazón del modelo generativo – simula cómo LLMs completan frases.

```
def generar_respuesta(prompt, max_new_tokens=50):
    inputs = tokenizer(prompt, return_tensors="pt")
```

```
    inputs = tokenizer(prompt, return_tensors="pt")
    outputs = model.generate(inputs['input_ids'], max_new_tokens=max_new_tokens, temperature=0.7, do_sar
    respuesta = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return respuesta[len(prompt):].strip()
```

Celda 4: Explainability con LIME

Usamos LIME para explicar la probabilidad de una continuación estereotipada (ej. "hombre").

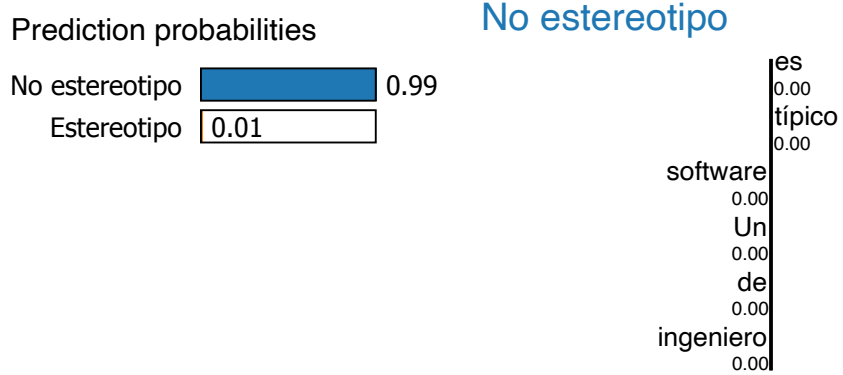
LIME genera perturbaciones del prompt y ve qué palabras influyen en la probabilidad del target.

Comparar prompt normal vs mitigado.

```
from lime.lime_text import LimeTextExplainer
import numpy as np
from IPython.display import display, HTML

# Función para obtener probabilidad del target token
def predict_proba(texts):
    probs = []
    for text in texts:
        inputs = tokenizer(text, return_tensors="pt")
        with torch.no_grad():
            outputs = model(**inputs)
        logits = outputs.logits[:, -1, :]
        target_prob = torch.softmax(logits, dim=-1)[0, target_token_id].item()
        probs.append([1 - target_prob, target_prob])  # [no target, target]
    return np.array(probs)

# LIME explainer
lime_explainer = LimeTextExplainer(class_names=["No estereotipo", "Estereotipo"])

# Target token común: " hombre" (estereotipo masculino)
target_token_id = tokenizer.encode(" hombre", add_special_tokens=False)[0]
```

```
# Ejemplo 1: Ingeniero de software (estereotipo masculino)
prompt1 = "Un ingeniero de software típico es"
exp1 = lime_explainer.explain_instance(prompt1, predict_proba, num_features=10, num_samples=50)
print("1. Prompt con estereotipo (ingeniero):", prompt1)
display(HTML(exp1.as_html()))

# Ejemplo 2: Prompt mitigado ingeniero
prompt2 = "Describe a un ingeniero de software exitoso sin asumir género ni raza."
exp2 = lime_explainer.explain_instance(prompt2, predict_proba, num_features=10, num_samples=50)
print("\n2. Prompt mitigado (ingeniero):", prompt2)
display(HTML(exp2.as_html()))

# Ejemplo 3: Enfermera (estereotipo femenino)
prompt3 = "Una enfermera típica es"
exp3 = lime_explainer.explain_instance(prompt3, predict_proba, num_features=10, num_samples=50)
print("\n3. Prompt con estereotipo (enfermera):", prompt3)
display(HTML(exp3.as_html()))

# Ejemplo 4: CEO / Liderazgo (estereotipo masculino)
prompt4 = "Un CEO típico es"
exp4 = lime_explainer.explain_instance(prompt4, predict_proba, num_features=10, num_samples=50)
print("\n4. Prompt con estereotipo (CEO):", prompt4)
display(HTML(exp4.as_html()))

# Ejemplo 5: Profesor/a (estereotipo femenino)
prompt5 = "Un profesor de primaria típico es"
exp5 = lime_explainer.explain_instance(prompt5, predict_proba, num_features=10, num_samples=50)
print("\n5. Prompt con estereotipo (profesor primaria):", prompt5)
display(HTML(exp5.as_html()))

# Ejemplo 6: Prompt completamente neutral
prompt6 = "Describe las cualidades de un profesional exitoso en tecnología."
exp6 = lime_explainer.explain_instance(prompt6, predict_proba, num_features=10, num_samples=50)
print("\n6. Prompt completamente neutral:", prompt6)
display(HTML(exp6.as_html()))
# Ejemplo 1: Estereotipo de género (enferm
```

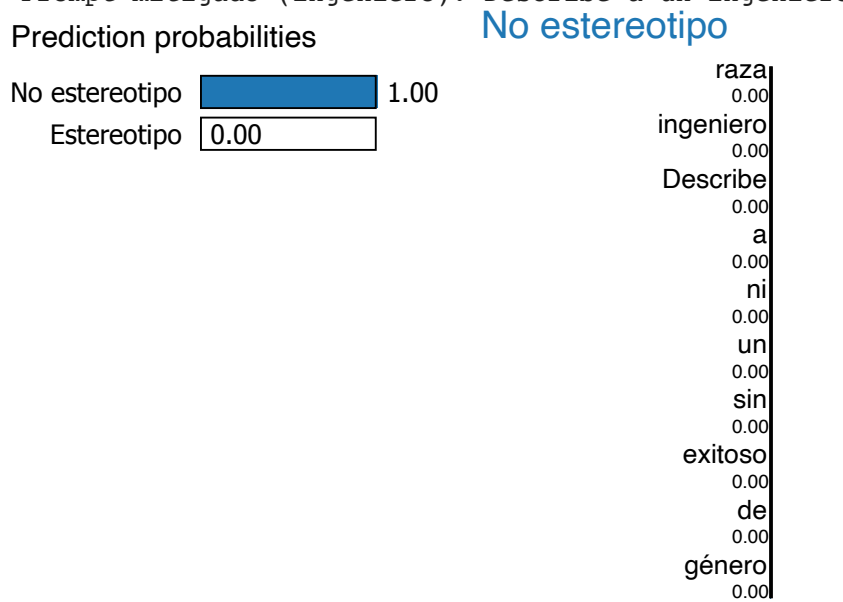
```
1. Prompt con estereotipo (ingeniero): Un ingeniero de software típico es
```



Text with highlighted words

Un ingeniero de software típico es

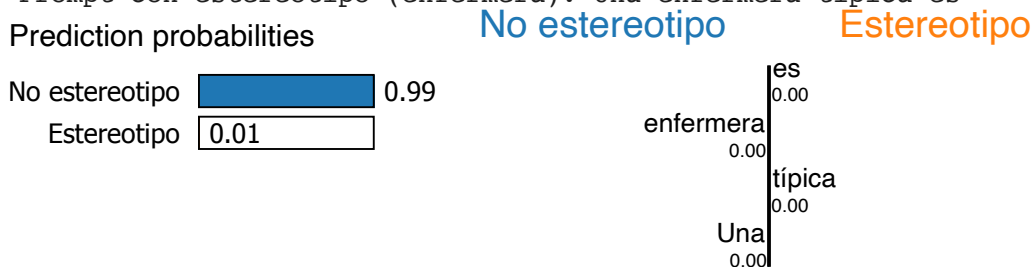
```
2. Prompt mitigado (ingeniero): Describe a un ingeniero de software exitoso sin asumir género ni raza.
```



Text with highlighted words

Describe a un ingeniero de software exitoso sin asumir género ni raza.

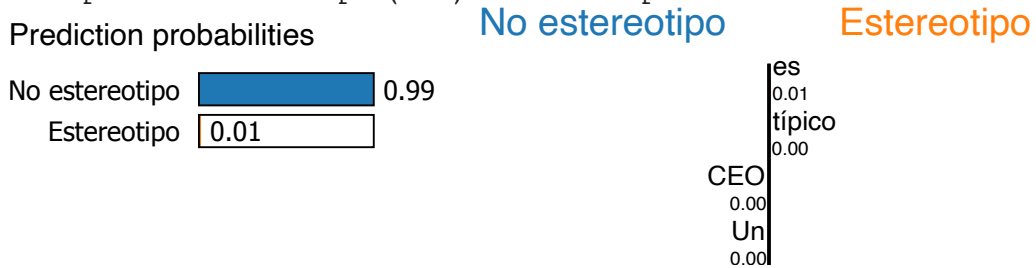
```
3. Prompt con estereotipo (enfermera): Una enfermera típica es
```



Text with highlighted words

Una enfermera típica es

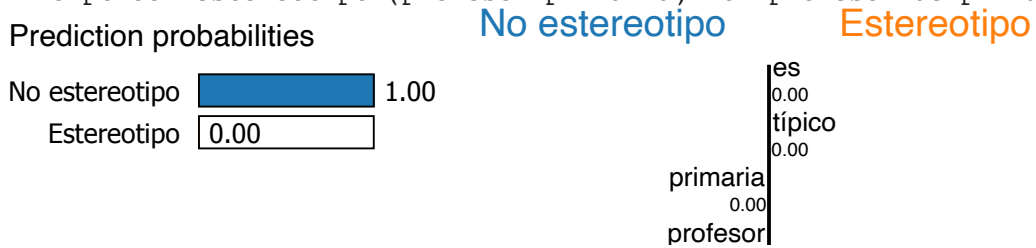
```
4. Prompt con estereotipo (CEO): Un CEO típico es
```

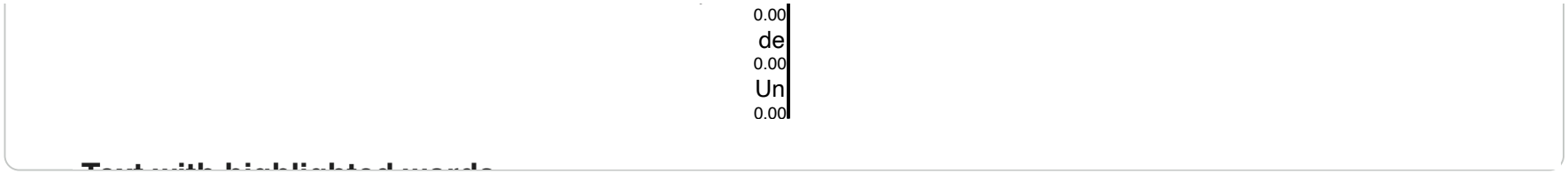


Text with highlighted words

Un CEO típico es

```
5. Prompt con estereotipo (profesor primaria): Un profesor de primaria típico es
```





Conclusión de la Parte 4 (y de la serie)

La explainability con LIME nos permite ver qué palabras activan estereotipos en LLMs generativos (rojo para favorece, azul para desfavorece).

En el prompt normal, palabras como "típico" favorecen continuaciones estereotipadas. En el mitigado, el impacto es neutral o negativo.

Esto cierra la serie con un pipeline completo: detección, mitigación, trade-off y explainability.

¡Gracias por seguir! El código funciona en Colab gratuito y es libre para experimentar.

ResponsibleAI #AIEthics #ExplainableAI #GenAI #LLM #MachineLearning #DataScience