

Ciencia de Datos aplicada a la IA Generativa

¿Qué es el Procesamiento de Lenguaje Natural (NLP)?

El **Procesamiento de Lenguaje Natural** (NLP, por sus siglas en inglés: *Natural Language Processing*) es una rama de la **inteligencia artificial** y la **lingüística computacional** que se enfoca en **hacer que las máquinas entiendan, interpreten, generen y trabajen con el lenguaje humano** (texto o voz).


¿Cómo se relaciona con la IA Generativa?

La **IA generativa de texto** (como ChatGPT, Claude, Gemini, etc.) se **basa completamente en técnicas de NLP**, pero lleva el procesamiento al siguiente nivel:

- Utiliza **modelos de lenguaje** (como los transformers) entrenados con **grandes volúmenes de texto**.
- Aprende patrones lingüísticos, gramaticales y semánticos.
- Genera **contenido nuevo y coherente** en forma de texto (respuestas, artículos, historias, resúmenes, código, etc.).

En otras palabras: **la IA generativa de texto es una aplicación avanzada del NLP**.


¿Y qué tiene que ver la Ciencia de Datos?

 **Ciencia de Datos:** Proporciona las herramientas para recolectar, limpiar, analizar y preparar datos textuales.



 **NLP:** Se especializa en entender y trabajar con lenguaje humano, usando datos procesados.



 **IA Generativa (texto):** Usa modelos de NLP entrenados con ciencia de datos para crear texto nuevo, coherente y útil.

Rol del NLP en el Pipeline

NLP (Procesamiento de Lenguaje Natural) es el conjunto de técnicas que permite transformar el texto escrito en una forma que la máquina pueda entender y procesar.

Todas las etapas que describiremos (limpieza, normalización, stemming/lematización, tokenización y vectorización) son parte de este proceso. La IA generativa se basa en estos elementos para construir modelos que, a partir de una entrada textual, generen contenido nuevo.

Aclaraciones

Un **corpus** es un conjunto grande y estructurado de textos que se utiliza para analizar, entrenar y evaluar modelos de procesamiento de lenguaje natural (NLP)

El preprocesamiento se aplica de manera similar tanto a los datos de entrenamiento como al prompt. Esto garantiza que el modelo trate de forma coherente la información, manteniendo la misma representación del lenguaje en ambos casos.

Flujo del Pipeline

Imaginemos que el prompt es:

"Contame una historia de un perro"

A. Extracción de Datos

Ejemplo:

Recopilamos artículos, libros o conversaciones en los que se hablan historias sobre perros. Este corpus crudo (conjunto extenso y no estructurado de textos) puede contener textos con etiquetas HTML, emojis, o formatos inconsistente.

B. Preprocesamiento de Texto

Esta fase transforma el texto crudo en un formato limpio y uniforme.

1. Limpieza y Normalización

Ejemplo del Texto Crudo:

```
"<p>¡Hola! ¿Te gustaría leer una historia sobre un <b>perro</b> muy especial? 😊</p>"
```

- **Limpieza:**
 - Se eliminan etiquetas HTML:
→ "¡Hola! ¿Te gustaría leer una historia sobre un perro muy especial? 😊"

- Se quitan los emojis (o se reemplazan por palabras, si se desea capturar el sentimiento):
→ "¡Hola! ¿Te gustaría leer una historia sobre un perro muy especial?"
- Se eliminan caracteres especiales redundantes (en este ejemplo, si se decide que los signos de exclamación o interrogación no aportan al análisis, se retiran):
→ "Hola te gustaria leer una historia sobre un perro muy especial"
- **Normalización:**
 - Pasar todo a minúsculas:
→ "hola te gustaria leer una historia sobre un perro muy especial"
 - Eliminar espacios extra o tabulaciones (en este caso, ya quedó bastante limpio).

2. Transformación Lingüística (Stemming o Lematización)

El objetivo es unificar variantes de una misma palabra:

- **Ejemplo de Lematización:**
 - "gustaria" se transforma a "gustar".
 - "leyendo", si apareciera, se transforma a "leer".

Aplicando lematización podríamos obtener:

→ "hola gustar leer una historia sobre un perro muy especial"

Nota: En algunos casos se usa **stemming**. Por ejemplo, "historias" se podría reducir a "histori", pero la lematización suele ser preferida porque preserva mejor el significado (por ejemplo, dejando "historia" en vez de "histori").

3. Tokenización

Dividimos el texto normalizado en unidades menores llamadas tokens:

- **Ejemplo:**

Del texto "hola gustar leer una historia sobre un perro muy especial",
→ Se obtiene una lista de tokens:
["hola", "gustar", "leer", "una", "historia", "sobre", "un", "perro", "muy", "especial"]

En este ejemplo, se hace tokenización por palabra. Si se usara tokenización por subpalabras (por ejemplo, Byte-Pair Encoding), la palabra "especial" podría dividirse en partes como ["espec", "ial"], según el vocabulario entrenado.

4. Conversión a IDs Numéricos y Vectorización (Embeddings)

Cada token se mapea a un número único según un vocabulario predefinido y, luego, a un vector que captura la semántica:

- **Ejemplo:**

Supongamos que el vocabulario asigna los siguientes IDs:

- "hola" → 101
- "gustar" → 205
- "leer" → 310
- "una" → 55
- "historia" → 478
- "sobre" → 312
- "un" → 42
- "perro" → 567
- "muy" → 89
- "especial" → 123

La secuencia se transforma a:

[101, 205, 310, 55, 478, 312, 42, 567, 89, 123]

Luego, cada uno de estos números se convierte en un vector de características. Por ejemplo, el embedding para "perro" podría ser:

[0.15, -0.30, 0.87, ...]

y para "historia", algo similar con otras coordenadas.

Estos vectores permiten al modelo comprender relaciones semánticas, de manera que vectores cercanos en este espacio implican conceptos relacionados (por ejemplo, "perro", "mascota" y "animal").

C. Modelado y Entrenamiento

Una vez que tenemos el texto en forma de secuencia de vectores (embeddings), se usan como entrada para entrenar un modelo generativo (por ejemplo, un Transformer como GPT).

- **Ejemplo de Funcionamiento:**

Cuando el modelo recibe la secuencia vectorial correspondiente al prompt "contame una historia de un perro", utiliza el embedding de "perro" y de las demás palabras para entender el contexto.

Durante el entrenamiento, el modelo aprende que esta secuencia está asociada a textos narrativos, donde usualmente se describen aventuras, emociones y detalles sobre el animal.

Luego, al generarse texto, el modelo "explora" el espacio latente formado por estos embeddings para construir una historia coherente y creativa.