Vocabulary Challenge

Para este challenge necesitamos entender que como paso preliminar del NLP que es una disciplina dentro de la inteligencia artificial que permite a las computadoras analizar, comprender y generar lenguaje humano, necesitamos tener un per-procesamiento del texto en el cual es importante recordar que, en última instancia, los modelos de aprendizaje profundo (Deep Learning) son funciones diferenciables que únicamente pueden operar con estructuras numéricas, como los tensores.

Por esta razón, es necesario transformar los datos de entrada desde texto a valores numéricos y, posteriormente, convertir esos valores en tensores mediante un procedimiento conocido como Vectorización de Texto. Este proceso se lleva a cabo a través de los siguientes pasos:

1. Estandarización: Se transforma el texto a un formato que facilite su procesamiento, lo que implica convertir todos los caracteres a minúsculas, eliminar los espacios en blanco innecesarios y suprimir la puntuación.
2. Tokenización: Se segmenta el texto en unidades mínimas (tokens), que pueden ser caracteres individuales, palabras o grupos de palabras.
3. Cálculo de Frecuencia: Se determina la cantidad de veces que aparece cada token en el texto, lo que a menudo requiere la construcción de un vocabulario.
4. Codificación Vectorial o Indexación: Cada token se convierte en un vector numérico que representa su posición dentro del vocabulario o su relación con otros términos en el corpus analizado.

Este procedimiento es fundamental para permitir que los modelos de aprendizaje profundo procesen y comprendan el lenguaje natural de manera eficiente.

En el challenge procesamos el libro de “Los Miserables” por Victor Hugo, por lo que con ayuda de python se realizó lo siguiente:

1. Se crea una función que extrae el texto del PDF asegurando únicamente tomar texto, convertimos el resultado en un DataFrame y lo guardamos en un archivo cdv:

```python
import pdfplumber
import pandas as pd
import re
import numpy as np
# from collections import defaultdict
from unidecode import unidecode

# --- Paso 1: Leer el PDF y extraer texto ---
def extract_text_from_pdf(pdf_path):
    text = []
    with pdfplumber.open(pdf_path) as pdf:
        for page in pdf.pages:
            extracted_text = page.extract_text()
            if extracted_text:  # Asegurar que haya texto en la página
                text.append(extracted_text)
    return text

pdf_path = "/Users/sebastianmerino/Downloads/Los-miserables.pdf"
book_text = extract_text_from_pdf(pdf_path)

# Convertir a DataFrame y guardar como CSV
df = pd.DataFrame({"text": book_text})
#df.to_csv("los_miserables_raw.csv", index=False)
```

2. Se crea una función que convierte en minúsculas, elimina acentos, espacios y caracteres alfanuméricos, se aplica la función a cada fila del DataFrame.

```
# --- Paso 2: Estandarizar y limpiar el texto ---
def standardize_text(text):
    text = text.lower()                    # Convertir a minúsculas
    text = unidecode(text)                  # Remover caracteres especiales
    text = re.sub(r'\W+', ' ', text)        # Remover puntuación/ no alfanumericos
    return text.strip()

# Aplicar la función para cada fila del DataFrame
df["text_cleaned"] = df["text"].apply(lambda x: standardize_text(x) if isinstance(x, str) else "")
# df.to_csv("los_miserables_cleaned.csv", index=False)
```

3. Se crea la función que nos ayuda a construir el vocabulario y se calculan las frecuencias, creamos un diccionario donde se almacenen las palabras, y se compara cada una de las nuevas palabras contra el diccionario del paso anterior, si se encuentra se aumenta la frecuencia de esta palabra en una unidad, y si no la encuentra se agrega al diccionario. Lo convertimos a DataFrame y ordenamos por frecuencia para conocer las palabras que aparecen más en el texto, finalmente guardamos esto en parquet.

```
# --- Paso 4: Estadisticas ---
total_words = sum(vocab_dict.values())  # Total de palabras en el libro
unique_words = len(vocab_dict)  # Palabras únicas en el vocabulario
most_frequent_words = vocab_df.head(100)  # Top 100 palabras más frecuentes
words_above_threshold = vocab_df[vocab_df["frequency"] > 100]  # Palabras que aparecen +100 veces
words_with_freq_1 = vocab_df[vocab_df["frequency"] == 1]  # Palabras que aparecen solo una vez

# Resultados
print(f"Total de palabras en el libro: {total_words}")
print(f"Palabras únicas en el vocabulario: {unique_words}")
print("\nTop 10 palabras más frecuentes:")
print(most_frequent_words.head(10))

print(f"\nPalabras que aparecen más de 100 veces: {len(words_above_threshold)}")
print(f"Palabras que aparecen solo una vez: {len(words_with_freq_1)}")
```

4. Se calculan las estadísticas del total de palabras del libro, palabras únicas en el diccionario de vocabulario, las 100 palabras más frecuentes, las palabras que aparecen sólo una vez.

```python
# --- Paso 3: Construir el vocabulario y calcular frecuencias ---
def build_vocabulary(text_series):
    vocabulary = {}  # Almacena las palabras y sus frecuencias

    for text in text_series.dropna():
        words = text.split()
        for word in words:
            if word in vocabulary:
                vocabulary[word] += 1  # Aumenta la frecuencia de palabras ya existentes
            else:
                vocabulary[word] = 1  # Añade nuevas palabras

    return vocabulary

# Crear el vocabulario a partir del texto limpio
vocab_dict = build_vocabulary(df["text_cleaned"])

# Convertir a DataFrame
vocab_df = pd.DataFrame(list(vocab_dict.items()), columns=["word", "frequency"])
vocab_df = vocab_df.sort_values(by="frequency", ascending=False)  # Ordenar por frecuencia

# Guardar en parquet
vocab_df.to_parquet("/Users/sebastianmerino/Downloads/Ejercicio_Los_miserables/vocabulary.parquet")
```

5. Finalmente se tienen las estadísticas calculadas

```
Total words in book: 109947
Unique words in vocabulary: 13461

Top 10 Most Frequent Words:
   word  frequency
10   de       5328
24   la       3918
86  que       3818
56   el       3394
19    y       3123
20   en       2836
23    a       2490
6    se       1681
53   un       1601
79   no       1498

Words appearing more than 100 times: 105
Words appearing only once: 7493
```