ACTIVIDAD 1

Taller:

Preparación del Entorno

pip install nltk matplotlib

Importación de Bibliotecas

Se importarán las bibliotecas necesarias para el análisis:

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import matplotlib.pyplot as plt
```

Carga de Texto

Se cargará un texto de ejemplo para el análisis, utilizando un corpus disponible en nltk, como nltk.corpus.gutenberg:

```
# Descargar el corpus Gutenberg si no está disponible
nltk.download('gutenberg')

# Cargar un texto de ejemplo
```

ACTIVIDAD 1

texto = nltk.corpus.gutenberg.raw('shakespeare-hamlet.txt')

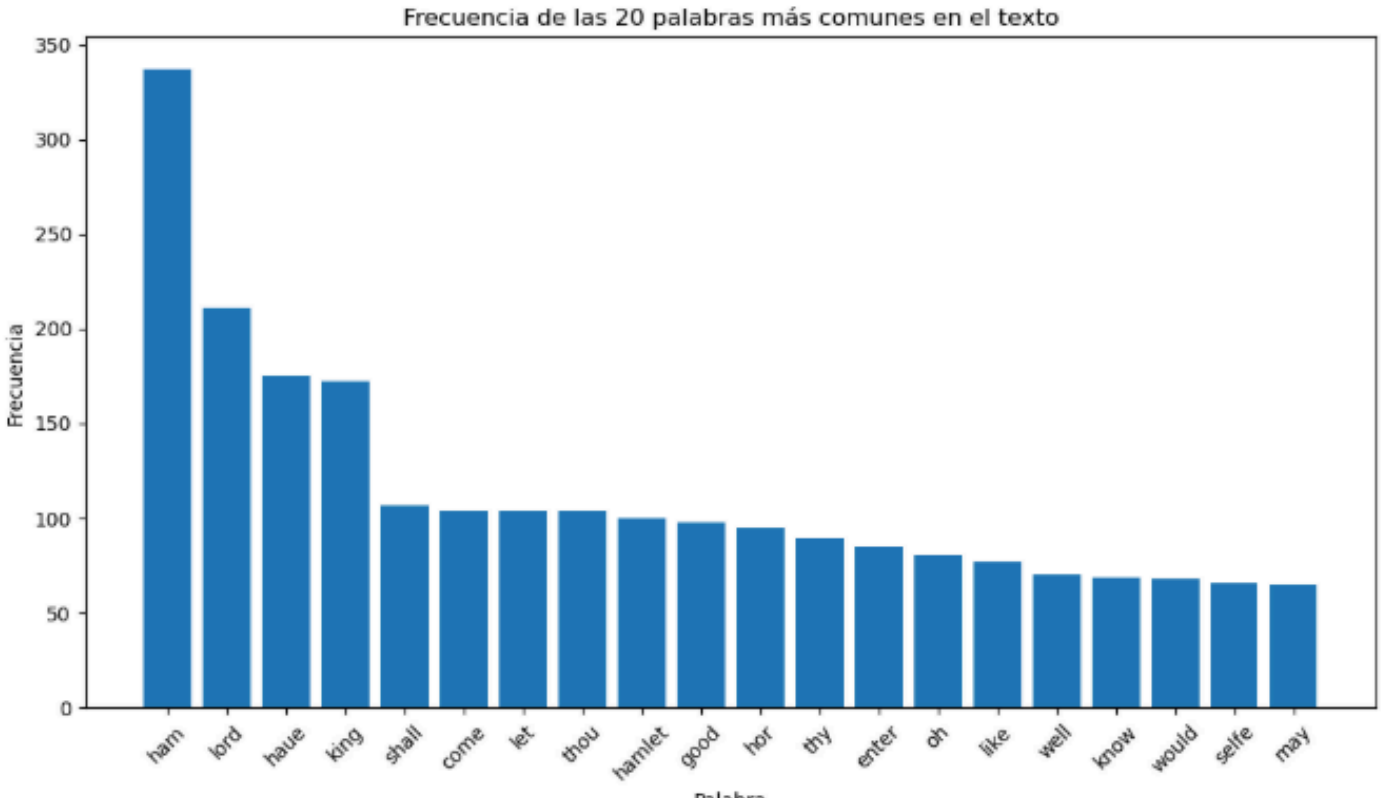
Tokenización y Frecuencia de Palabras

Se llevará a cabo la tokenización del texto y se calculará la frecuencia de palabras. Además, se eliminarán las palabras vacías (stop words) para un análisis más preciso:

```
# Tokenización
tokens = word_tokenize(texto)

# Eliminación de stop words
stop_words = set(stopwords.words('english'))
tokens = [word.lower() for word in tokens if word.isalpha() and word.lower() not in stop_words]

# Cálculo de la frecuencia de palabras
frecuencia_palabras = nltk.FreqDist(tokens)
```

```
# Obtener las 20 palabras más comunes

palabras_comunes = frecuencia_palabras.most_common(20)
palabras, frecuencias = zip(*palabras_comunes)

# Visualización de las palabras más comunes

plt.figure(figsize=(10, 6))

plt.bar(palabras, frecuencias)

plt.title('Frecuencia de las 20 palabras más comunes en el texto')

plt.xlabel('Palabra')

plt.ylabel('Frecuencia')

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()
```

Esto deberá arrojar lo siguiente:



Se muestra un diagrama de barras que contiene la distribución de frecuencia de los datos. Para este caso las 20 primeras palabras más comunes, para este caso en la distribución de frecuencias de palabras en el texto, se utilizan principalmente dos bibliotecas: NLTK (Natural Language Toolkit) y Matplotlib. NLTK proporciona una amplia gama de herramientas y recursos para el procesamiento del lenguaje natural en Python, incluyendo funciones para tokenizar texto, calcular la frecuencia de palabras y eliminar palabras vacías (stop words). Matplotlib es una biblioteca de visualización que nos permite crear gráficos de barras, como el utilizado en

este caso para mostrar las palabras más comunes y sus frecuencias. Juntas, estas bibliotecas proporcionan las herramientas necesarias para analizar la distribución de frecuencias de palabras en un texto y visualizar los resultados de manera efectiva.

Las stop words (palabras vacías) son palabras comunes que suelen eliminarse del texto durante el procesamiento del lenguaje natural debido a que no aportan un significado importante para el análisis. Estas palabras son muy frecuentes en un idioma dado, pero por lo general carecen de información semántica útil. Ejemplos de stop words en inglés incluyen "the", "is", "and", "are", "in", "to", entre otras.

La eliminación de stop words es una técnica común en el preprocesamiento de texto, ya que ayuda a reducir el ruido y el tamaño del vocabulario, lo que a su vez puede mejorar la precisión de ciertas tareas de procesamiento del lenguaje natural, como la clasificación de textos o la recuperación de información. Sin embargo, la lista de stop words puede variar dependiendo del contexto y del objetivo del análisis.

Estadísticas en Textos

Se calcularán algunas estadísticas básicas sobre el texto, como la longitud promedio de las palabras y la diversidad léxica:

```
# Longitud promedio de las palabras
longitudes_palabras = [len(word) for word in tokens]
longitud_promedio = sum(longitudes_palabras) / len(longitudes_palabras)
print('Longitud promedio de las palabras:', longitud_promedio)

# Diversidad léxica

diversidad_lexica = len(set(tokens)) / len(tokens)
print('Diversidad léxica:', diversidad_lexica)
```

Las estadísticas más comunes de esto son:

1. Longitud promedio de las palabras:

Longitud promedio de las palabras= $\frac{\sum_{i=1}^{n} longitud(w_i)}{n}$;

2. Diversidad léxica:

Diversidad léxica = V/n
donde:

- n es el número total de palabras en el texto.
- longitud(wi) es la longitud de la palabra wiwi.
- V es el número de palabras únicas en el texto.

El anterior scripts, imprimirá lo siguiente:

Longitud promedio de las palabras: 5.302612764851164

Diversidad léxica: 0.2894189522942935 4

Otro análisis exclusivo usando textos es el de identificación de género, a continuación se usa la base de datos del titanic,con dicho ejemplo.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import spacy
```

```
# Cargar el modelo en español de spaCy

nlp = spacy.load("es_core_news_sm")

# Cargar los datos del Titanic
train_data = pd.read_csv("titanic/train.csv")
test_data = pd.read_csv("titanic/test.csv")

# Combinar los datos de entrenamiento y prueba

all_data = pd.concat([train_data, test_data], ignore_index=True)

# Función para determinar el género y la edad aproximada

def categorize_gender_age(name):
 name = name.lower()
 if 'mr.' in name:
    return 'Hombre'
 elif 'mrs.' in name or 'miss.' in name:
    return 'Mujer'
 elif 'master.' in name or 'boy' in name or 'baby' in
```

name:

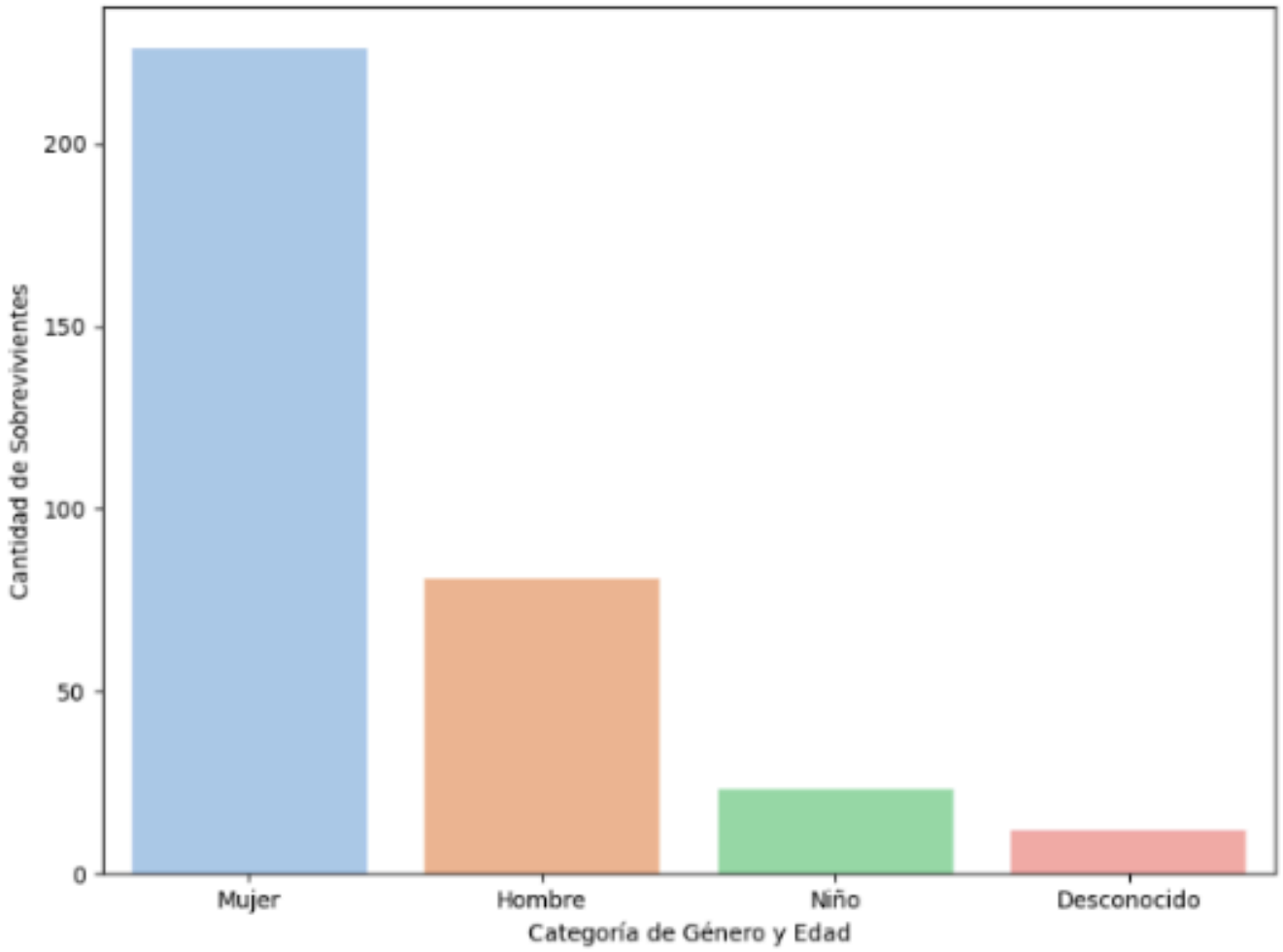
```
    return 'Niño'
    return 'Desconocido'

# Aplicar la función a cada fila
all_data['Gender_Age_Category'] =
all_data['Name'].apply(categorize_gender_age)

# Contar la cantidad de sobrevivientes por categoría
de género y edad
survivors_count = all_data[all_data['Survived'] == 1]
['Gender_Age_Category'].value_counts()

# Visualización de la cantidad de sobrevivientes por
género y edad
plt.figure(figsize=(8, 6))
sns.countplot(x='Gender_Age_Category',
data=all_data[all_data['Survived'] == 1], palette='pastel')

#plt.title('Cantidad de Sobrevivientes por Género y
Edad')
plt.xlabel('Categoría de Género y Edad')
plt.ylabel('Cantidad de Sobrevivientes')
plt.tight_layout()
plt.show()
```

```
# Imprimir los resultados

print("Cantidad de Sobrevivientes por Género y Edad:")
print(survivors_count)
```

Para complementar el análisis de los datos del Titanic, se aplicaron técnicas básicas de Procesamiento del Lenguaje Natural (NLP) para extraer información útil de los nombres de los pasajeros. Primero, se definió una función que busca palabras clave en los nombres, como "Mr.", "Mrs.", "Miss.", "Master.", "Boy" y "Baby", para determinar el género y la edad aproximada de cada pasajero. Estas palabras clave se utilizaron como indicadores para clasificar a los pasajeros en categorías de género y edad: Hombre, Mujer y Niño. Luego, se aplicó esta función a cada nombre en el conjunto de datos para asignar una categoría de género y edad a cada pasajero. Esto permitió realizar un análisis más detallado de la distribución de los sobrevivientes en función de su género y edad. (Ver siguiente figura)

```
# Imprimir los resultados

print("Cantidad de Sobrevivientes por Género y Edad:")
print(survivors_count)
```

Para complementar el análisis de los datos del Titanic, se aplicaron técnicas básicas de Procesamiento del Lenguaje Natural (NLP) para extraer información útil de los nombres de los pasajeros. Primero, se definió una función que busca palabras clave en los nombres, como "Mr.", "Mrs.", "Miss.", "Master.", "Boy" y "Baby", para determinar el género y la edad aproximada de cada pasajero. Estas palabras clave se utilizaron como indicadores para clasificar a los pasajeros en categorías de género y edad: Hombre, Mujer y Niño. Luego, se aplicó esta función a cada nombre en el conjunto de datos para asignar una categoría de género y edad a cada pasajero. Esto permitió realizar un análisis más detallado de la distribución de los sobrevivientes en función de su género y edad. (Ver siguiente figura)



Esta representa la cantidad de sobrevivientes para un conjunto de datos del titanic, con las palabras reservadas según el dataset, se identificaron la frecuencia de dichos datos.