Actividad NLP - 1.0 - Creación de diccionario

Mario Alberto Castañeda Martínez - A01640152

La primera parte de la actividad será hacer un contador de palabras, haciendo un primero un histograma, para después eliminar preposiciones y generar una nube de palabras. La segunda parte consiste en crear un diccionario de palabras y corrección del texto usando cálculo de distancia de strings.

El texto que se usará será el libro "The Great Gatsby" de Scott Fitzgerald, el cual fue descargado con ayuda de la página web "Project Gutemberg". Cabe mencionar que el archivo donde se encuentra el libro es un .txt.

Se importan las librerías necesarias:

```
from google.colab import drive
drive.mount('/content/drive')
%cd "/content/drive/MyDrive/CONCENTRACION AI/data"
```

```
Mounted at /content/drive
/content/drive/MyDrive/CONCENTRACION AI/data
```

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import string
```

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

Se obtiene el .txt con el libro y se aplica tokenize para separar cada palabra del texto:

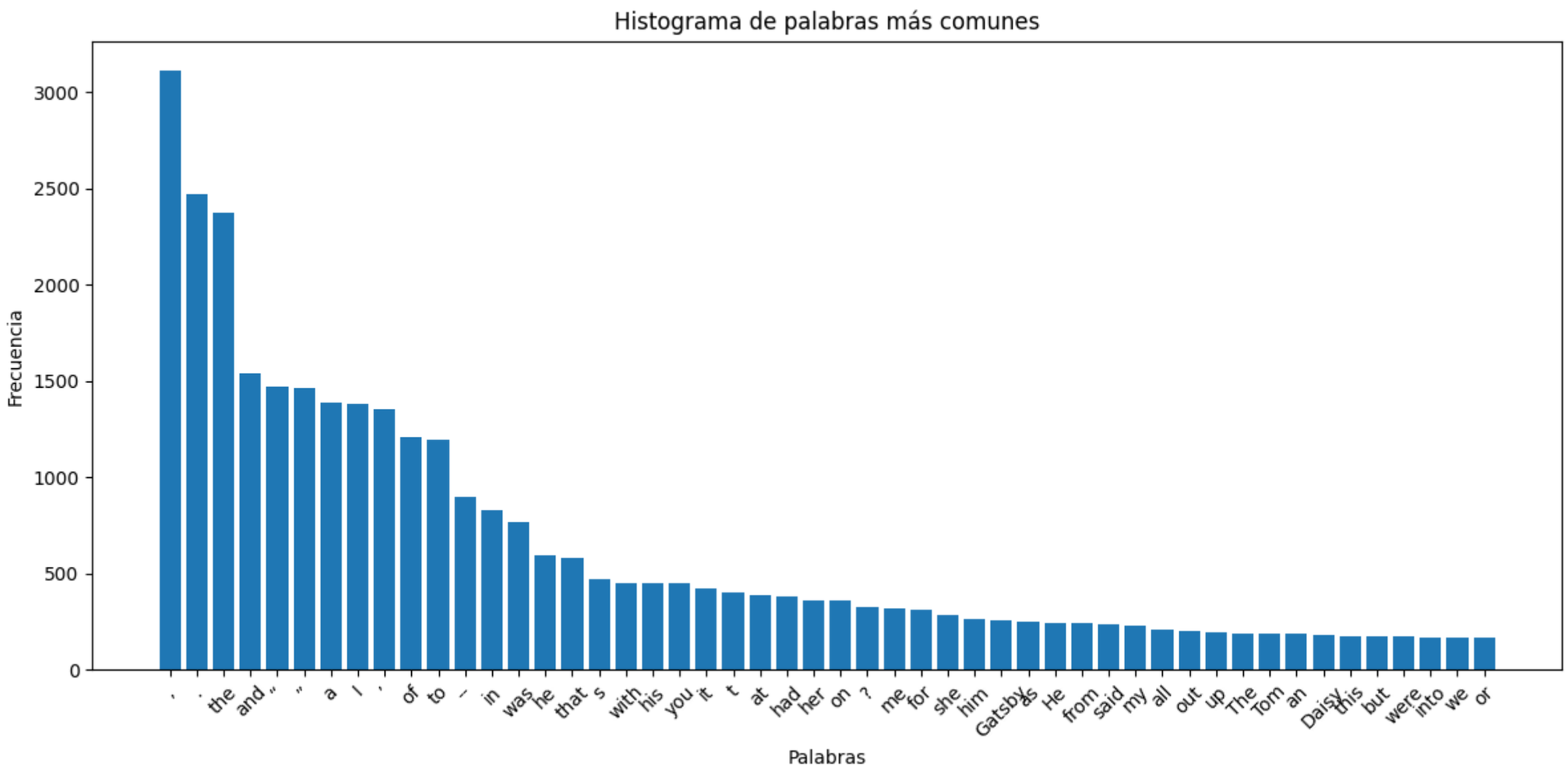
```
with open('great_gatsby.txt', 'r', encoding='utf-8') as file:
    texto = file.read()
    palabras = word_tokenize(texto)
```

Después de haber separado las palabras, se contará la frecuencia de cada palabra en el libro:

```
frecuencia = FreqDist(palabras)
```

Teniendo la frecuencia de las palabras en el texto, se va a visualizar el conteo de las 50 palabras más comunes en el texto, con el objetivo de tener una vista general del conteo de palabra del texto:

```
mas_comunes = frecuencia.most_common(50)
palabras, frecuencias = zip(*mas_comunes)
# Crear el histograma
plt.figure(figsize=(12, 6))
plt.bar(palabras, frecuencias)
plt.title('Histograma de palabras más comunes')
plt.xlabel('Palabras')
plt.ylabel('Frecuencia')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Al observar el histograma, se observan muchos signos de puntuación al igual que prepocisiones, por lo que se procederá a eliminarlas para volver a generar un histograma del conteo:

```
with open('great_gatsby.txt', 'r', encoding='utf-8') as file:
    texto = file.read()
```

Se convierten en minúsculas las palabras y se eliminan los signos de puntuación del texto:

```
texto = texto.lower()


translator = str.maketrans('', '', string.punctuation)
texto = texto.translate(translator)
```

Se separan las palabras con tokenize:

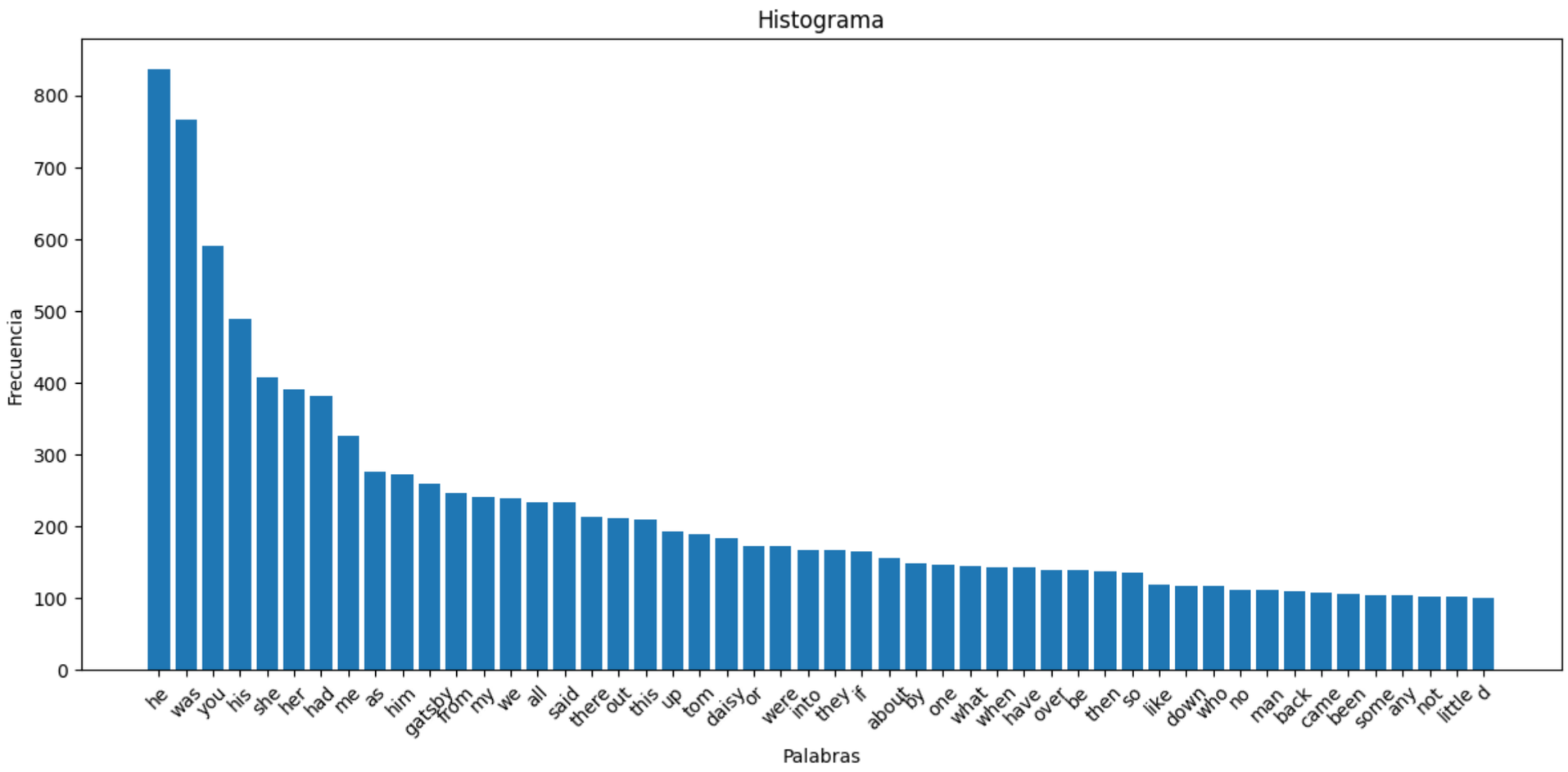
```
palabras = word_tokenize(texto)
```

Se eliminan palabras como prepocisiones y otros signos:

```
stop_words = set(['“','”','’','i','--','s','t','?','a', 'an', 'the', 'in', 'on', 'at', 'of', 'to', 'for', 'with', 'and', 'but', 'is', 'it', 'that'])
```

Se vuelve a generar un histograma de las palabras más usadas en el texto:

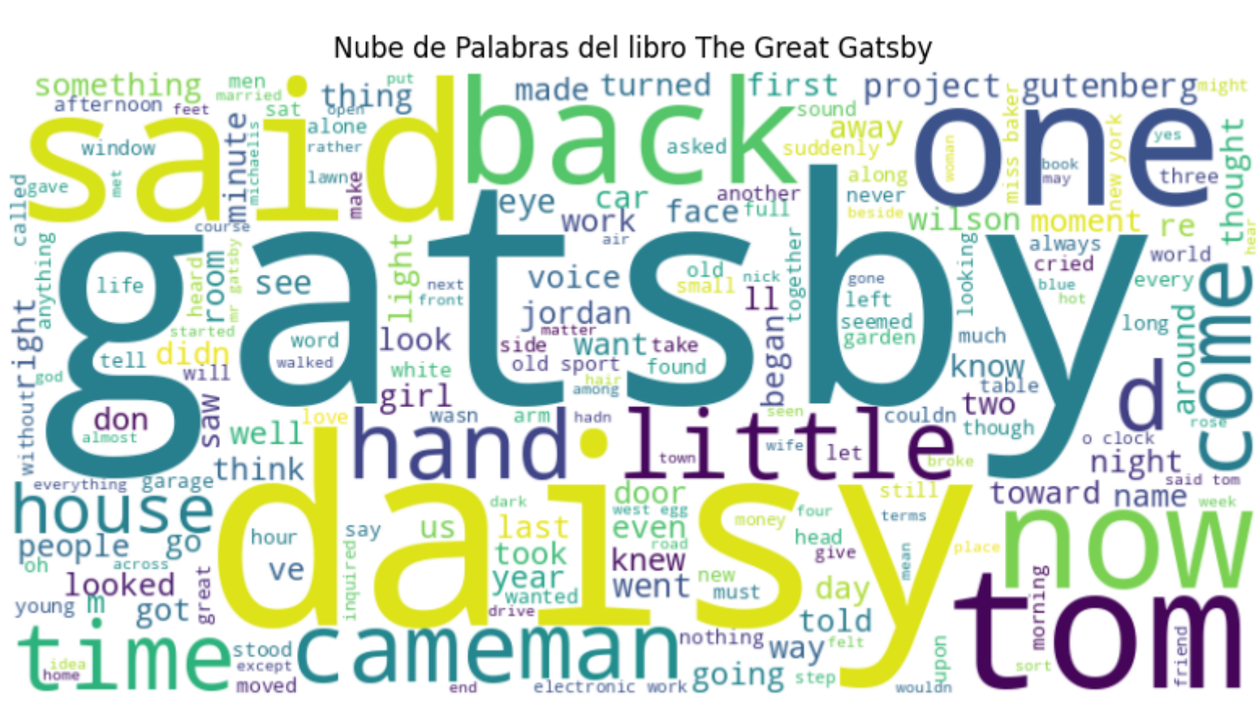
```
filtro_palabras = [word for word in palabras if word not in stop_words]

frecuencia = FreqDist(filtro_palabras)

mas_comunes = frecuencia.most_common(50)

words, frequencies = zip(*mas_comunes)
plt.figure(figsize=(12, 6))
plt.bar(words, frequencies)
plt.title('Histograma')
plt.xlabel('Palabras')
plt.ylabel('Frecuencia')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



En este nuevo histograma quitando las preposiciones y signos de puntución, se observa que las palabras más comunes, son palabras del tipo he,she,you, etc. Se observan igualmente nombres de los personajes principales (gatsby, tom, daisy) además de algunos verbos conjugados y sin conjugar.

Ahora se procede a realizar una nube de palabras del texto del libro:

```
from wordcloud import WordCloud
filtro_texto = ' '.join(filtro_palabras)
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(filtro_texto)
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Nube de Palabras del libro The Great Gatsby')
plt.show()
```



En la nube de palabras generada, se observa que las palabras mostradas, no muestran las palabras del tipo he,she,you, was, etc, y en su lugar se muestran palabras más relevantes para el libro y la historia, tales como nombres de los personajes, algunos verbos, entre otras. Considero

que la nube muestra un gran resumen o vistazo general del libro, donde se pueden ver personajes y algunas palabras que le dan forma a la historia contada en el libro.

Creación de diccionario de palabras

Se procede ahora a hacer un diccionario de palabras del libro:

Se genera un diccionario donde se visualizan todas las palabras presentes en el libro y su frecuencia:

```
with open('great_gatsby.txt', 'r', encoding='utf-8') as file:
    text = file.read()

words = text.split()

word_count = {}
for word in words:
    word = word.strip('.,?!";:()[]{}“”')
    word = word.lower()

    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1
```

```
print(word_count)
```

```
{'\ufeffthe': 1, 'project': 88, 'gutenberg': 31, 'ebook': 13, 'of': 1232, 'the': 2571, 'great': 30, 'gatsby': 193, 'this': 209, 'is': 125, 'for': 334, 'use': 20, 'anyone':
```

Ahora se hace una corrección del texto usando la distancia de Levenshtein y respecto al diccionario:

```
pip install Levenshtein
```

```
Collecting Levenshtein
  Downloading Levenshtein-0.23.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (169 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 169.4/169.4 kB 3.2 MB/s eta 0:00:00
Collecting rapidfuzz<4.0.0,>=3.1.0 (from Levenshtein)
  Downloading rapidfuzz-3.4.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 44.8 MB/s eta 0:00:00
Installing collected packages: rapidfuzz, Levenshtein
Successfully installed Levenshtein-0.23.0 rapidfuzz-3.4.0
```

```
import Levenshtein



def find_closest_word(word, word_count):
    min_distance = float('inf')
    closest_word = word

    for dict_word in word_count:
        distance = Levenshtein.distance(word, dict_word)
        if distance < min_distance:
            min_distance = distance
            closest_word = dict_word

    return closest_word

corrected_text = []
for word in words:
    corrected_word = find_closest_word(word, word_count)
    corrected_text.append(corrected_word)

corrected_text = ' '.join(corrected_text)

print(corrected_text)
```

```
the project gutenberg ebook of the great gatsby this ebook is for the use of anyone anywhere in the united states and most other parts of the world at no cost and with almo
```