Ejercicio: Tokenización en Python

Ejercicio: Tokenización en Python

```
# Importar bibliotecas necesarias
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize

# Definir función para tokenizar texto por palabras
def tokenize_words(text):
    return word_tokenize(text)

# Definir función para tokenizar texto por frases
def tokenize_sentences(text):
    return sent_tokenize(text)

# Texto de ejemplo
texto_ejemplo = "Este es un ejemplo de texto. Incluye varias frases y palabras diferentes."

# Tokenizar texto por palabras
tokens_palabras = tokenize_words(texto_ejemplo)
print("Tokens por palabras:", tokens_palabras)
```

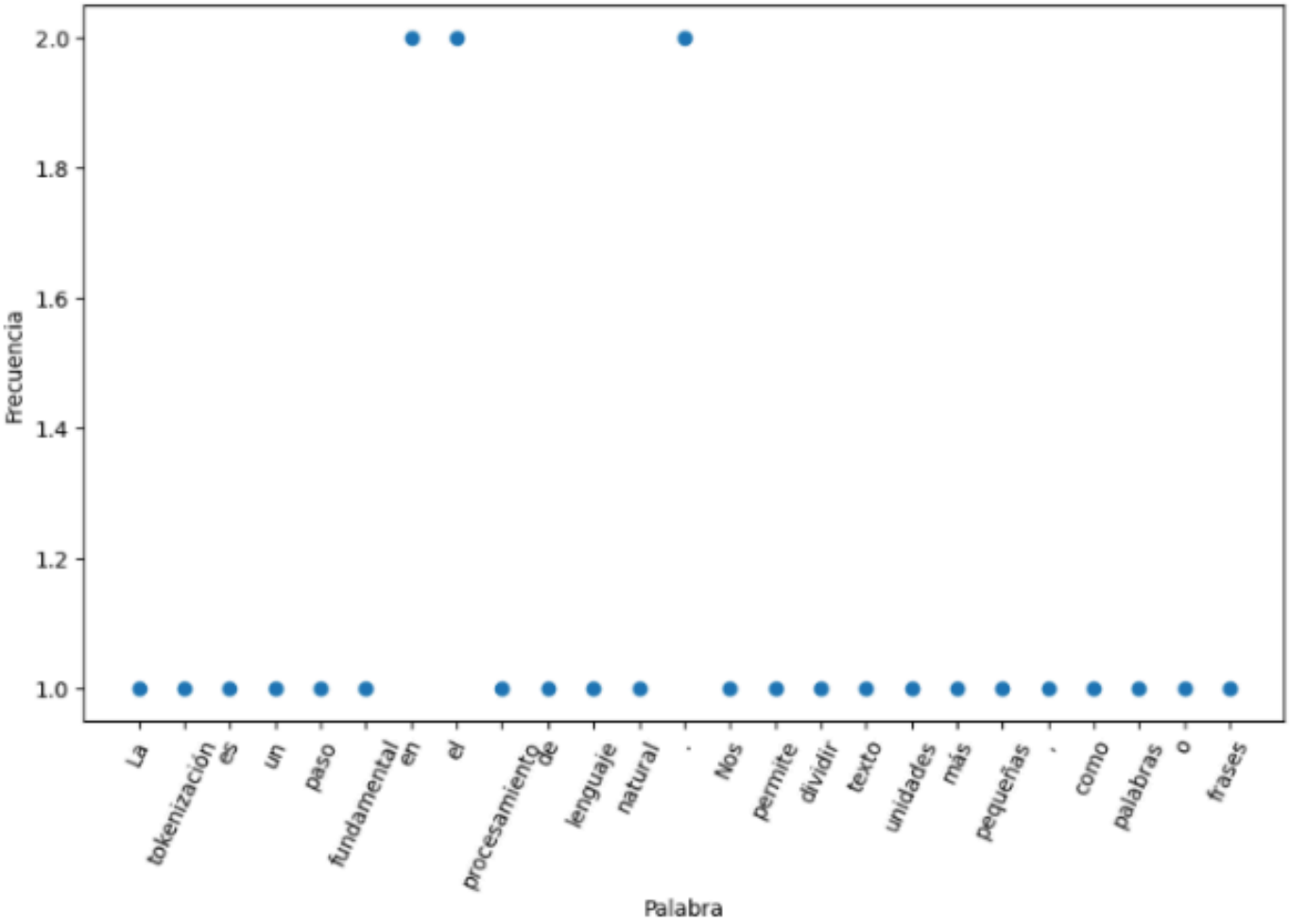
```
i# Tokenizar texto por frases
tokens_frases = tokenize_sentences(texto_ejemplo)
print("Tokens por frases:", tokens_frases)
```

Esto anterior, permite tokenizar por palabas y por frases, vea lo que arrojaTokens por palabras: ['Este', 'es', 'un', 'ejemplo', 'de', 'texto', '.', 'Incluye', 'varias', 'frases', 'y', 'palabras', 'diferentes', '.']
Tokens por frases: ['Este es un ejemplo de texto.', 'Incluye varias frases y palabras diferentes.']
Python muestra cómo tokenizar un texto utilizando la biblioteca NLTK. La función **tokenize_words** se utiliza para dividir el texto en tokens individuales por palabras, mientras que la función **tokenize_sentences** se utiliza para dividir el texto en tokens por frases.

Otra forma de visualization es la cantidad de repeticiones de las palabras tokenizdas, esto permite tener un rango de visualization más grande, para dicho análisis.

```
# Paso 1: Importar las bibliotecas necesarias
import nltk
import matplotlib.pyplot as plt
```

```
from nltk.probability import FreqDist
nltk.download('punkt')

# Paso 2: Definir el texto a tokenizar
texto = "La tokenización es un paso fundamental en el procesamiento de lenguaje natural. Nos permite dividir el texto en unidades más pequeñas, como palabras o frases."

# Paso 3: Tokenizar el texto
tokens = nltk.word_tokenize(texto)

# Paso 4: Calcular la frecuencia de cada palabra
frecuencia = FreqDist(tokens)

# Paso 5: Crear un diagrama de dispersión de frecuencia de palabras
plt.figure(figsize=(10, 6))
plt.scatter(frecuencia.keys(), frecuencia.values())
plt.title('Diagrama de Dispersión de Frecuencia de Palabras')
plt.xlabel('Palabra')
plt.ylabel('Frecuencia')
plt.xticks(rotation=45) # Rotar las etiquetas del eje x para mayor claridad
plt.show()
```



El diagrama de dispersión de frecuencia de palabras, como una herramienta en el análisis de datos de texto, proporciona una visualización efectiva de la distribución y frecuencia de las palabras en un texto tokenizado. Cada punto en el diagrama representa una palabra, donde su posición en el eje x indica la palabra y su frecuencia se representa en el eje y.

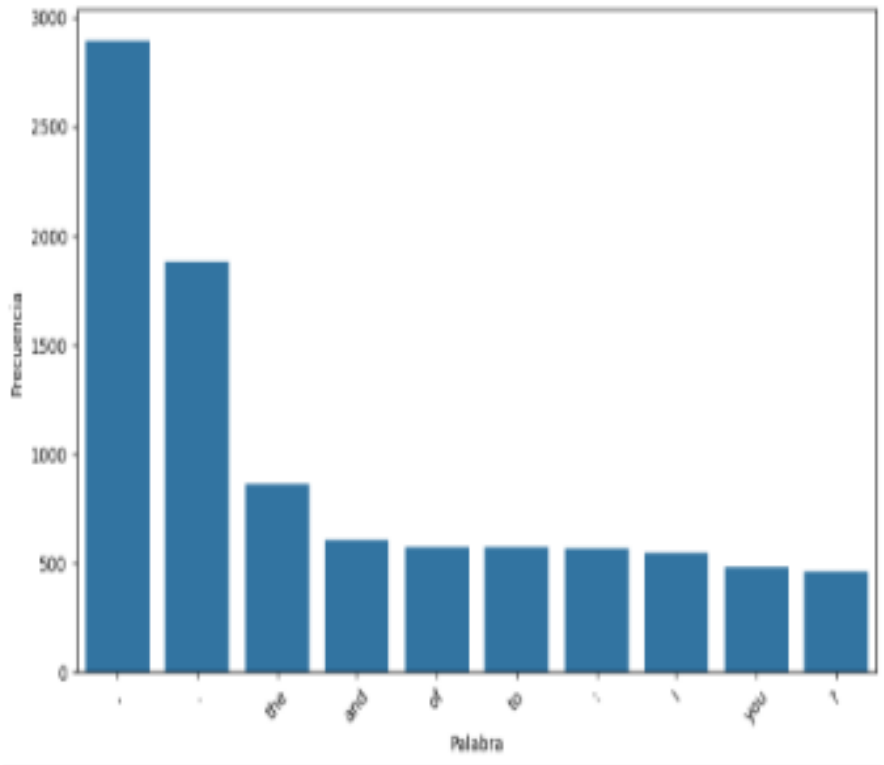
Esta representación permite una rápida identificación de las palabras más frecuentes y las menos frecuentes en el texto, lo que resulta crucial para comprender la composición y las tendencias del texto analizado. Este enfoque visual facilita la extracción de información relevante y la identificación de patrones importantes en el análisis de datos de texto.

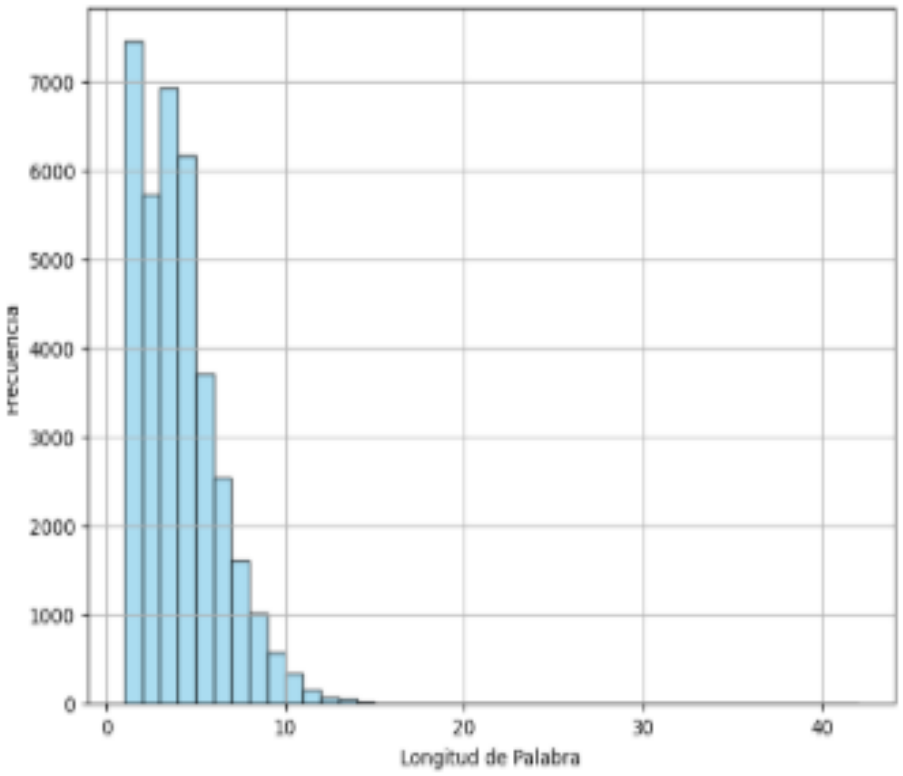
La tokenización es ampliamente utilizada en el análisis estadístico, lo que permite la generación de gráficos analíticos como histogramas de longitudes de palabras, gráficos de barras de frecuencias, boxplots, entre otros. A continuación se muestra un script que utiliza el corpus de NLTK, una biblioteca de procesamiento de lenguaje natural, además de abordar la probabilidad con la librería NLTK.

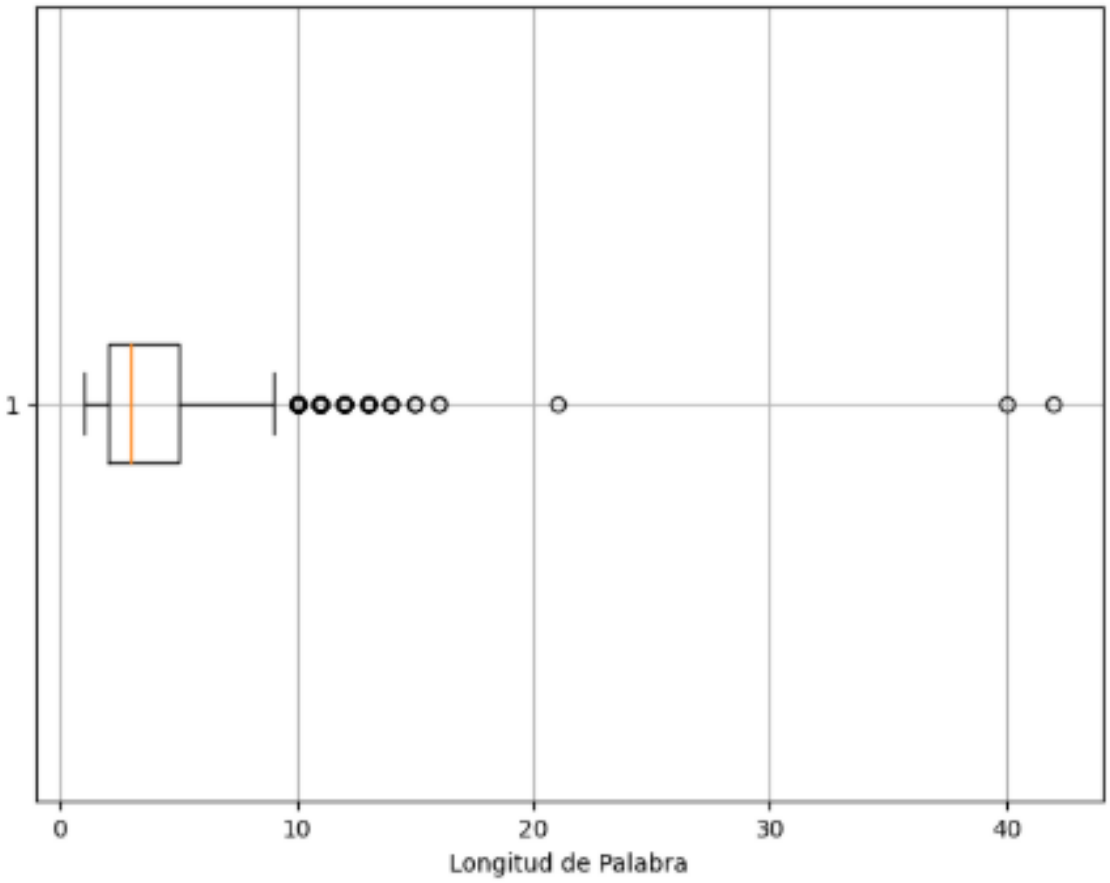
```
import nltk
from nltk.corpus import gutenberg
from nltk.tokenize import word_tokenize
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.probability import FreqDist
from wordcloud import WordCloud
```

```
# Descargar los recursos necesarios de NLTK
nltk.download('punkt')
nltk.download('gutenberg')

# Obtener un ejemplo de texto del corpus de NLTK
texto = gutenberg.raw('shakespeare-hamlet.txt')

# Tokenización y obtención de longitudes de palabras
tokens = word_tokenize(texto)
longitudes = [len(token) for token in tokens]

# Histograma de Longitudes de Palabras
plt.figure(figsize=(8, 6))
plt.hist(longitudes, bins=range(min(longitudes), max(longitudes) + 1), alpha=0.7, color='skyblue', edgecolor='black')
plt.title('Histograma de Longitudes de Palabras')
plt.xlabel('Longitud de Palabra')
plt.ylabel('Frecuencia')
plt.grid(True)
plt.show()

# Gráfico de Barras de Frecuencia de Palabras
frecuencia = FreqDist(tokens)
palabras_comunes = frecuencia.most_common(10)
```

```
plt.figure(figsize=(10, 6))
sns.barplot(x=[word[0] for word in palabras_comunes],
y=[word[1] for word in palabras_comunes])


plt.title('Palabras más Frecuentes')
plt.xlabel('Palabra')
plt.ylabel('Frecuencia')
plt.xticks(rotation=45)
plt.show()


# Boxplot de Longitudes de Palabras
plt.figure(figsize=(8, 6))
plt.boxplot(longitudes, vert=False)
plt.title('Boxplot de Longitudes de Palabras')
plt.xlabel('Longitud de Palabra')
plt.grid(True)
plt.show()
```


3000
2500
2000
1500
1000
500
0
Frecuencia
the
and
of
to
Palabra


7000
6000
5000
4000
3000
2000
1000
0
Frecuencia
0
10
20
30
40
Longitud de Palabra



El análisis gráfico realizado detalla las características lingüísticas del texto analizado. El histograma de longitudes de palabras revela la distribución de la longitud de las palabras, destacando la tendencia en el uso de palabras de ciertas longitudes. Por otro lado, el gráfico de barras de frecuencia de palabras resalta las palabras más comunes y su importancia relativa en el texto, permitiendo identificar las palabras dominantes y su frecuencia de aparición. El boxplot de longitudes de palabras ofrece una comprensión de la variabilidad en las longitudes de las palabras y la presencia de posibles valores atípicos.