NLTK (Natural Language Toolkit) es una biblioteca de Python ampliamente utilizada en el procesamiento del lenguaje natural (NLP). Ofrece una variedad de funciones y herramientas para trabajar con texto, incluidos archivos de texto (.txt) y otros tipos de archivos. Para archivos de texto (.txt), NLTK proporciona varias funciones y herramientas útiles, incluyendo:

1. Tokenización: Divide un texto en unidades más pequeñas, como palabras o frases.
2. Lematización y stemming: Reducción de palabras a su forma base (lemas o raíces).
3. Análisis de frecuencia de palabras: Conteo de la frecuencia de cada palabra en un texto.
4. Part-of-speech tagging: Etiquetado gramatical de palabras en un texto.
5. Análisis de sentimientos: Determinación del tono o la polaridad del texto.
6. Procesamiento de n-gramas: Extracción de secuencias de palabras contiguas de longitud n.
7. Clasificación de texto: Categorización automática de textos en diferentes clases. Para trabajar con archivos de texto en NLTK, generalmente se sigue un flujo de trabajo que implica cargar el texto desde un archivo .txt, preprocesarlo (limpiarlo, tokenizarlo, etc.), y luego aplicar las herramientas NLP según sea necesario. Aquí tienes una función específica de NLTK y cómo se usa: Función: nltk.word_tokenize()

- Propósito: Tokeniza un texto en palabras individuales.
- Sintaxis: `nltk word_tokenize(texto)`

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

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

Ejemplos:

```
import nltk
from nltk.tokenize import word_tokenize


texto = "NLTK es una biblioteca de procesamiento de lenguaje natural muy útil."
tokens = word_tokenize(texto)
print(tokens)
```

```
['NLTK', 'es', 'una', 'biblioteca', 'de', 'procesamiento', 'de', 'lenguaje', 'natural', 'muy', 'útil', '.']
```

```
from nltk.tokenize import sent_tokenize
texto = "¡Hola! ¿Cómo estás? Espero que estés bien."
oraciones = sent_tokenize(texto)
print(oraciones)
```

```
['¡Hola!', '¿Cómo estás?', 'Espero que estés bien.']
```

```
from nltk.stem import WordNetLemmatizer
lemmatizador = WordNetLemmatizer()
palabra = "hola"
print(lemmatizador.lemmatize(palabra))  # Output: 'hola'
```

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
palabra = "corriendo"
print(stemmer.stem(palabra))  # Output: 'corriendo'
```

```
corriendo
```

```
from nltk.probability import FreqDist
palabras = nltk.word_tokenize(texto)
frecuencia = FreqDist(palabras)
print(frecuencia.most_common(5))
```

```
[('¡Hola', 1), ('!', 1), ('¿Cómo', 1), ('estás', 1), ('?', 1)]
```

```
import nltk
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

```
from nltk import pos_tag
palabras = nltk.word_tokenize(texto)
etiquetas = pos_tag(palabras)
print(etiquetas)
```

```
[('¡Hola', 'NN'), ('!', '.'), ('¿Cómo', 'NN'), ('estás', 'NN'), ('?', '.'), ('Espero', 'NNP'), ('que', 'NN'), ('estés', 'NN'), ('bien',
```

```
import nltk
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
True
```

```
from nltk.sentiment import SentimentIntensityAnalyzer
analisador_sentimientos = SentimentIntensityAnalyzer()
texto = "Me siento muy feliz hoy."
sentimiento = analisador_sentimientos.polarity_scores(texto)
print(sentimiento)
```

```
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
```

```
from nltk.util import ngrams
tokens = nltk.word_tokenize(texto)
bigramas = list(ngrams(tokens, 2))
print(bigramas)
```

```
[('Me', 'siento'), ('siento', 'muy'), ('muy', 'feliz'), ('feliz', 'hoy'), ('hoy', '.')]
```

```
from nltk.classify import NaiveBayesClassifier
from nltk.classify.util import apply_features

def extraer_caracteristicas(texto):
    # Aquí puedes definir cómo extraer características del texto
    # Por ejemplo, puedes tokenizar el texto y contar las ocurrencias de palabras
    palabras = texto.lower().split()
    return dict([(palabra, True) for palabra in palabras])

datos_entrenamiento = [('Esto es genial', 'positivo'), ('Esto es terrible', 'negativo')]
caracteristicas_entrenamiento = [(extraer_caracteristicas(texto), etiqueta) for texto, etiqueta in datos_entrenamiento]
modelo = NaiveBayesClassifier.train(caracteristicas_entrenamiento)

texto = "Esto es increíble"
caracteristicas_texto = extraer_caracteristicas(texto)
clasificacion = modelo.classify(caracteristicas_texto)
print(clasificacion)
```

```
positivo
```

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
from nltk.corpus import stopwords
palabras = nltk.word_tokenize(texto)
stopwords_esp = set(stopwords.words('spanish'))
palabras_filtradas = [palabra for palabra in palabras if palabra.lower() not in stopwords_esp]
print(palabras_filtradas)
```

```
['increíble']
```

```
import nltk
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
True
```

```
from nltk import ne_chunk
palabras = nltk.word_tokenize(texto)
etiquetas = pos_tag(palabras)
entidades = ne_chunk(etiquetas)
print(entidades)
```

```
(S (GPE Esto/NNP) es/VBZ increíble/JJ)
```

Este código importa NLTK, carga la función `word_tokenize ()`, tokeniza el texto proporcionado y luego imprime los tokens resultantes. Para otros tipos de archivos, como archivos PDF, HTML, XML, etc., NLTK no proporciona herramientas específicas para leer esos archivos. Sin embargo, puedes usar otras bibliotecas de Python para leer esos archivos y luego usar NLTK para procesar el texto extraído. Por ejemplo, puedes usar `pdfminer` para leer archivos PDF o bibliotecas como `BeautifulSoup` para analizar archivos HTML/XML. Una vez que hayas extraído el texto, puedes aplicar las funciones de NLTK según sea necesario.