



# UNIVERSIDAD DE COLIMA

## Facultad de Ingenieria Mecanica y Electrica

### Ingenieria en Computación Inteligente

## PROCESAMIENTO DE LENGUAJE NATURAL

## NLTK

### Alba Beth-birai López Aguilar 20186885

Funciones de la libreria NLTK :

1. Análisis de corpus:

- **nltk.corpus:** Módulo que proporciona acceso a una colección de corpus de texto pre-anotados, como el Brown Corpus o el Corpus de Sentido Común.
- **Concordance():** Genera concordancias para una palabra o frase específica, mostrando las apariciones de la palabra o frase en el corpus.
- **collocation\_list():** Genera una lista de colocaciones para una palabra o frase específica, mostrando las palabras que suelen aparecer juntas en el corpus.

2. Análisis de sentimientos:

- **nltk.sentiment:** Módulo para realizar análisis de sentimientos, como la clasificación de opiniones y la detección de sarcasmo.

- **SentimentIntensityAnalyzer():** Analiza la intensidad del sentimiento en una pieza de texto.
- **VaderSentimentIntensityAnalyzer():** Analiza la intensidad del sentimiento en una pieza de texto utilizando el conjunto de datos de VADER (Valence Aware Dictionary and sEntiment Reasoner).

### 3. Traducción de idiomas:

- **nltk.translate:** Módulo para traducir texto de un idioma a otro.
- **GoogleTranslate():** Traductor de texto que utiliza la API de traducción de Google.
- **Moses():** Traductor de texto que utiliza el motor de traducción estadística Moses.

### 4. Generación de texto:

- **nltk.text:** Módulo para realizar tareas de generación de texto, como la generación de oraciones o la creación de historias.
- **NgramLanguageModel():** Genera texto utilizando un modelo de lenguaje n-gram.
- **SequenceLM():** Genera texto utilizando un modelo de lenguaje de secuencia a secuencia.

### 5. Visualización de datos:

- **nltk.visualize:** Módulo para visualizar datos de procesamiento del lenguaje natural.
- **FreqDistPlot():** Crea un gráfico de distribución de frecuencias para una palabra o frase específica.
- **discourse\_plot():** Crea un gráfico de discurso que muestra la distribución de las palabras en un texto.

In [ ]: *##EJEMPLO DE LOS USOS DE LAS FUNCIONES MENCIONADAS*

```
import nltk
from nltk.corpus import brown

# Acceder a una muestra del corpus de Brown
texto = brown.words(categories='news')[:50]
print(texto)
```

In [ ]: **from nltk.text import Text**

```
# Convertir la lista de palabras en un objeto Text
texto = Text(brown.words())

# Generar concordancias para la palabra "government"
texto.concordance("government")

# Generar una lista de colocaciones para la palabra "good"
collocations = texto.collocation_list(num=20, window_size=2)
print(collocations)
```

```
from nltk.sentiment import SentimentIntensityAnalyzer

# Crear un analizador de intensidad de sentimiento
sia = SentimentIntensityAnalyzer()

# Analizar el sentimiento de una frase
frase = "I love this movie, it's amazing!"
sentimiento = sia.polarity_scores(frase)
print(sentimiento)
```

```
In [ ]: from googletrans import Translator

# Crear un objeto Translator
translator = Translator()

# Traducir una frase de inglés a español
frase_ingles = "Hello, how are you?"
traduccion_espanol = translator.translate(frase_ingles, src='en', dest='es')
print(traduccion_espanol.text)
```

Funciones dentro de la librería NLTK para archivos txt:

1. Lectura y escritura de archivos:

- `open()`: Abre un archivo de texto para lectura o escritura.
- `read()`: Lee todo el contenido de un archivo de texto.
- `readline()`: Lee una línea a la vez del archivo de texto.
- `write()`: Escribe contenido en un archivo de texto.
- `close()`: Cierra un archivo de texto.

2. Preprocesamiento de texto:

- `tokenize()`: Divide el texto en tokens (palabras, frases, etc.).
- `normalize()`: Normaliza el texto (conversión a minúsculas, eliminación de caracteres especiales, etc.).
- `stem()`: Reduce las palabras a su raíz (stemming).
- `lemmatize()`: Reduce las palabras a su forma léxica básica (lemmatización).

3. Análisis sintáctico:

- `pos_tag()`: Asigna etiquetas de parte del discurso a las palabras.
- `chunk_sents()`: Divide el texto en chunks (grupos de palabras con una función gramatical).
- `parse_tree()`: Crea un árbol de análisis sintáctico del texto.

4. Análisis semántico:

- `wordnet`: Acceso a la base de datos de WordNet para obtener información semántica sobre las palabras.

- nltk.sem: Módulo para realizar tareas de análisis semántico, como la desambiguación de sentidos y la inferencia de significado.

#### 5. Clasificación de texto:

- NaiveBayesClassifier(): Implementación del clasificador de Naive Bayes para clasificar textos.
- MaxEntClassifier(): Implementación del clasificador de máxima entropía para clasificar textos.
- SupportVectorMachineClassifier(): Implementación del clasificador de máquina de soporte vectorial para clasificar textos.

#### 6. Extracción de información:

- NamedEntityRecognizer(): Identifica entidades nombradas en el texto (personas, lugares, organizaciones, etc.).
- RelationExtractor(): Extrae relaciones entre entidades nombradas en el texto.

```
In [ ]: import nltk
from nltk.corpus import brown
from nltk.text import Text
from nltk.sentiment import SentimentIntensityAnalyzer
from googletrans import Translator

# Leer el archivo de texto
archivo = 'texto_extraido.txt'
with open(archivo, 'r') as f:
    texto = f.read()

#####
tokens = nltk.word_tokenize(texto) # Tokenizar el texto

#####
# Acceder al corpus de Brown
texto_brown = Text(brown.words())

#####
# Generar concordancias para la palabra "government"
print("Concordancias para la palabra 'government':")
texto_brown.concordance("government")

#####

collocations = texto_brown.collocation_list(num=20, window_size=2) # Generar una li
print("\nColocaciones para la palabra 'good':")
print(collocations)

#####
# Crear un analizador de intensidad de sentimiento
sia = SentimentIntensityAnalyzer()
```

```
#####
# Analizar el sentimiento del texto completo
sentimiento_texto = sia.polarity_scores(texto)
print("\nAnálisis de sentimiento del texto completo:")
print(sentimiento_texto)

#####
# Crear un objeto Translator
translator = Translator()

#####
# Traducir el texto de inglés a español
traduccion_espanol = translator.translate(texto, src='en', dest='es')
print("\nTexto traducido al español:")
print(traduccion_espanol.text)
```

Comandos de NLTK para otros tipos de archivos

#### 1. Archivos XML:

- nltk.corpus.reader.xml: Módulo para leer archivos XML.
- readtree(): Lee un archivo XML y lo representa como un árbol de elementos.
- xpath(): Aplica expresiones XPath a un árbol de elementos XML.

#### 2. Archivos JSON:

- nltk.json: Módulo para trabajar con archivos JSON.
- load(): Carga un archivo JSON como un objeto Python.
- dumps(): Convierte un objeto Python en una cadena JSON.

#### 3. Archivos de audio:

- nltk.corpus.reader.wav: Módulo para leer archivos de audio WAV.
- read(): Lee un archivo de audio WAV y lo representa como una matriz de datos.
- play(): Reproduce un archivo de audio WAV.

#### 4. Archivos de imagen:

- nltk.corpus.reader.vision: Módulo para trabajar con archivos de imagen (funcionalidad limitada).
- load\_image(): Carga una imagen desde un archivo.
- show(): Muestra una imagen.

Nota: Las funcionalidades de NLTK para archivos que no sean de texto plano son más limitadas en comparación con las herramientas especializadas en esos formatos. Se recomienda utilizar bibliotecas específicas para cada tipo de archivo si se requieren tareas más complejas.

```
In [ ]: import nltk
from nltk.corpus.reader import xml as xml_reader
from nltk.corpus.reader import wav as wav_reader
from nltk.corpus.reader import vision as vision_reader
import json

# Función para Leer un archivo XML y representarlo como un árbol de elementos
def leer_xml(file_path):
    tree = xml_reader.readtree(file_path)
    return tree

# Función para aplicar expresiones XPath a un árbol de elementos XML
def aplicar_xpath(tree, xpath_expr):
    result = tree.xpath(xpath_expr)
    return result

# Función para cargar un archivo JSON como un objeto Python
def cargar_json(file_path):
    with open(file_path, 'r') as f:
        data = json.load(f)
    return data

# Función para convertir un objeto Python en una cadena JSON
def convertir_a_json(objeto_python):
    json_string = json.dumps(objeto_python)
    return json_string

# Función para Leer un archivo de audio WAV y representarlo como una matriz de datos
def leer_audio(file_path):
    audio_data = wav_reader.read(file_path)
    return audio_data

# Función para reproducir un archivo de audio WAV
def reproducir_audio(audio_data):
    wav_reader.play(audio_data)

# Función para cargar una imagen desde un archivo
def cargar_imagen(file_path):
    image = vision_reader.load_image(file_path)
    return image

# Función para mostrar una imagen
def mostrar_imagen(image):
    vision_reader.show(image)

# Ejemplo de uso de las funciones
if __name__ == "__main__":
    # Leer archivo XML
    xml_file = 'ejemplo.xml'
    xml_tree = leer_xml(xml_file)
    print("Árbol XML:")
    print(xml_tree)

    # Aplicar expresión XPath al árbol XML
    xpath_expr = '/root/elemento/subelemento'
```

```
xpath_result = aplicar_xpath(xml_tree, xpath_expr)
print("\nResultado de la expresión XPath:")
print(xpath_result)

# Cargar archivo JSON
json_file = 'ejemplo.json'
json_data = cargar_json(json_file)
print("\nDatos del archivo JSON:")
print(json_data)

# Convertir datos Python a JSON
json_string = convertir_a_json(json_data)
print("\nDatos Python convertidos a JSON:")
print(json_string)

# Leer archivo de audio WAV
wav_file = 'ejemplo.wav'
audio_data = leer_audio(wav_file)

# Reproducir audio
print("\nReproduciendo archivo de audio...")
reproducir_audio(audio_data)

# Cargar imagen desde archivo
image_file = 'ejemplo.jpg'
imagen = cargar_imagen(image_file)

# Mostrar imagen
print("\nMostrando imagen...")
mostrar_imagen(imagen)
```