ANEXOS: PROCESAMIENTO DE LENGUAJE NATURAL(NLP) CON SCIPY Y NLTK

SESION 2 / SEMANA 2

Anexo 01 : Tarea Aprendizaje 03

Alumno: [aquí colocar su nombres y apellidos]

1. ¿En que casos puede usarse el Procesamiento de Lenguaje Natural (NLP)? (2 PTOS)

   El Procesamiento de Lenguaje Natural (NLP) puede usarse en casos como:

   - **Análisis de sentimientos:** para identificar emociones o actitudes en textos como reseñas o comentarios en redes sociales.
   - **Asistentes virtuales:** como chatbots o sistemas de respuesta automática que interactúan con los usuarios mediante lenguaje natural.
   - **Traducción automática:** servicios como Google Translate que convierten textos de un idioma a otro.
   - **Reconocimiento de voz:** convertir la voz en texto, como en asistentes personales tipo Alexa o Siri.

2. ¿Cuál es la diferencia entre SciPy y NLTK? (2 PTOS)

   - **SciPy:** Es una librería en Python enfocada en operaciones matemáticas, científicas y de ingeniería, como álgebra lineal, optimización y procesamiento de señales.
   - **NLTK:** Es una librería de Python para el Procesamiento de Lenguaje Natural (NLP) que proporciona herramientas para análisis de texto, tokenización, etiquetado gramatical y más.

   SciPy es para matemáticas y ciencia en general, mientras que NLTK está especializado en la manipulación y análisis de lenguaje humano.

3. ¿En que aplicaciones se usa el Modelado de Lenguaje y tokenización de texto? (2 PTOS)

   El modelado de lenguaje y la tokenización de texto se usan en:

   - **Motores de búsqueda:** para entender las consultas de los usuarios y proporcionar resultados más relevantes.
   - **Generación de texto:** en aplicaciones como GPT que generan texto coherente a partir de un input.
   - **Corrección gramatical:** en herramientas que analizan el texto para sugerir correcciones o mejoras, como Grammarly.
   - **Chatbots y asistentes virtuales:** para interpretar el lenguaje y responder de forma apropiada.

4. ¿Cuál es la diferencia entre paquete MachineLearning y Deep Learning? (2 PTOS)

   - **Machine Learning (ML):** es un subcampo de la inteligencia artificial que se enfoca en el desarrollo de algoritmos que pueden aprender de los datos y hacer predicciones o decisiones basadas en patrones detectados. Incluye técnicas como regresión lineal, árboles de decisión y SVM.
   - **Deep Learning (DL):** es un subcampo de Machine Learning que utiliza redes neuronales profundas (con muchas capas) para analizar datos de manera más compleja, como en reconocimiento de imágenes o procesamiento de lenguaje natural. Las redes neuronales permiten a DL manejar grandes volúmenes de datos y realizar tareas más avanzadas.

   Deep Learning es más complejo y requiere más datos y poder computacional que Machine Learning.

5. Implementar el siguiente programa en python que analiza los sentimientos de un texto “María está sentada en la playa, escuchando las olas del mar y el canto de las aves. Siente una profunda paz.” usando paquete NLTK y aplicando además técnicas de tokenización, etiquetado de partes, extracción de entidades. (12 PTOS). Rpta: Pantalla de Resultado. PD: no se olvide de descargar el paquete NLTK.

```
Import nltk

from nltk.tokenize import word_tokenize

from nltk import pos_tag, ne_chunk

from nltk.sentiment import SentimentIntensityAnalyzer

from nltk.corpus import stopwords

# Descargar recursos necesarios

nltk.dowload('punk')

nltk.dowload('punk_tab')

nltk.dowload('averaged_perceptron_tagger')

nltk.dowload('averaged_perceptron_tagger_eng')

nltk.dowload('maxent_ne_chunker')

nltk.dowload('maxent_ne_chunker_tab')

nltk.dowload('words')

nltk.dowload('vader_lexicon')

nltk.dowload('stopwords')

# Texto de ejemplo en español

text = "María está sentada en la playa, escuchando las olas del mar y el canto de las aves. Siente una profunda paz."

# Tokenización

tokens = word_tokenize(text, language='spanish')

print("Tokens:", tokens)

# Eliminación de stopwords

stop_words = set(stopwords.words('spanish'))

filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
```

```
print("\nFiltered Tokens:", filtered_tokens)
# Etiquetado de Partes del Discurso (POS Tagging)
pos_tags = pos_tag(filtered_tokens)
print("\nPOS Tags:", pos_tags)
#Extracción de Entidades Nombradas
named_entities = ne_chunk(pos_tags)
print("\nNamed Entities:")
for entity in named_entities:
if hasattr(entity, 'label' ):
print(entity.label(), ' '.join(c[0] for c in entity))
# Análisis de Sentimientos
sia = SentimentIntensityAnalyzer()
sentiment = sia.polarity_scores(text)
print("\nSentiment Analysis:", sentiment)
# Validación  Semántica y pragmática
if sentiment['compound'] > 0.5;
print("\nEl sentimiento de la oracion es positivo, lo cual coincide con el contexto de paz y tranquilidad en la playa.")
else:
print("\nEl sentimiento no es fuertemente positivo, lo que podria indicar una desalineacion en el contexto de paz.")
```

Welcome tarea3.py

tarea3.py > ...

```python
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag, ne_chunk
from nltk.sentiment import SentimentIntensityAnalyzer
from nltk.corpus import stopwords

# Descargar recursos necesarios
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger')
nltk.download('averaged_perceptron_tagger_eng')
nltk.download('maxent_ne_chunker')
nltk.download('maxent_ne_chunker_tab')
nltk.download('words')
nltk.download('vader_lexicon')
nltk.download('stopwords')

# Texto de ejemplo en español
text = "María está sentada en la playa, escuchando las olas del mar y el canto de las aves. Siente una profunda paz."

# Tokenización
tokens = word_tokenize(text, language='spanish')
print("Tokens:", tokens)

# Eliminación de stopwords
stop_words = set(stopwords.words('spanish'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
print("\nFiltered Tokens:", filtered_tokens)

# Etiquetado de Partes del Discurso (POS Tagging)
pos_tags = pos_tag(filtered_tokens)
print("\nPOS Tags:", pos_tags)

# Extracción de Entidades Nombradas
named_entities = ne_chunk(pos_tags)
print("\nNamed Entities:")
for entity in named_entities:
    if hasattr(entity, 'label'):
        print(entity.label(), ' '.join(c[0] for c in entity))

# Análisis de Sentimientos
sia = SentimentIntensityAnalyzer()
sentiment = sia.polarity_scores(text)
print("\nSentiment Analysis:", sentiment)

# Validación Semántica y Pragmática
if sentiment['compound'] > 0.5:
    print("\nEl sentimiento de la oración es positivo, lo cual coincide con el contexto de paz y tranquilidad en la playa.")
else:
    print("\nEl sentimiento no es fuertemente positivo, lo que podría indicar una desalineación con el contexto de paz.")
```