TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TLAXIACO

PRÁCTICA: CLASIFICADOR DE TEXTO BÁSICO UTILIZANDO LA LIBRERÍA SCIKIT-LEARN

YAEL DE JESUS SANTIAGO ORTIZ

PROFESOR: ING. JOSUÉ ISRAEL VÁZQUEZ MARTÍNEZ

INTELIGENCIA ARTIFICIAL

2U

8° US

ING SISTEMAS COMPUTACIONALES

13 MARZO DE 2025

INTRODUCCIÓN

El procesamiento de lenguaje natural (NLP, por sus siglas en inglés) es una rama de la inteligencia artificial que permite a las computadoras comprender, interpretar y producir texto humano. Uno de los enfoques fundamentales en el NLP es la clasificación de texto, que consiste en categorizar automáticamente fragmentos de texto en diferentes etiquetas predefinidas. En esta práctica, desarrollaremos un clasificador básico utilizando la librería de aprendizaje automático scikit-learn en Python.

OBJETIVO

El objetivo de esta práctica es implementar un modelo que clasifique comentarios de texto en dos categorías: positivo y negativo. Para ello, utilizaremos el método TF-IDF para procesar texto y el algoritmo de clasificación Naive Bayes multinomial.

MATERIALES

- ✧ Lenguaje de programación Python (versión 3.x).
- ✧ Librerías necesarias:
- ✧ NumPy.
- ✧ scikit-learn (que incluye TfidfVectorizer y MultinomialNB).

DESARROLLO

Preparación de los datos:

Se definió un conjunto de comentarios representativos en idioma español. Cada comentario fue etiquetado manualmente como "positivo" o "negativo" según su tono.

```
textos = [
    "Me encanta este producto, es increíble",  # Positivo
    "Es una porquería, no lo recomiendo",  # Negativo
```

```
    ...
]

etiquetas = ["positivo", "negativo", ...]
```

```python
# Datos de entrenamiento (Ejemplo: comentarios clasificados)
textos = [
    "Me encanta este producto, es increíble",  # Positivo
    "Es una porquería, no lo recomiendo",  # Negativo
    "Muy útil y funciona perfectamente",  # Positivo
    "No sirve para nada, pésima calidad",  # Negativo
    "Estoy muy feliz con esta compra",  # Positivo
    "Terrible, me arrepiento de comprarlo",  # Negativo
]

etiquetas = ["positivo", "negativo", "positivo", "negativo", "positivo", "negativo"]
```

Transformación del texto en vectores numéricos:

Utilizamos TfidfVectorizer para convertir los textos en una matriz numérica basada en la frecuencia de términos y su relevancia (TF-IDF). Este método asigna mayor peso a palabras que son importantes en un documento pero menos comunes en otros.

Construcción del pipeline:

Creamos un pipeline que combina la transformación de los datos con TF-IDF y el modelo de clasificación MultinomialNB para simplificar el flujo de trabajo.

```
modelo = make_pipeline(TfidfVectorizer(), MultinomialNB())
```

```python
# Convertir texto en números con TF-IDF y entrenar el modelo
modelo = make_pipeline(TfidfVectorizer(), MultinomialNB())
modelo.fit(textos, etiquetas)
```

Entrenamiento del modelo:

Entrenamos el modelo con los textos y etiquetas proporcionados. El clasificador aprende a diferenciar entre comentarios positivos y negativos basándose en las palabras presentes en cada texto.

```
modelo.fit(textos, etiquetas)
```

Predicción de nuevos datos:

Probamos el modelo con un comentario nuevo para predecir su etiqueta. Por ejemplo:

```
nuevo_texto = ["Este producto no es bueno, lo odio"]

prediccion = modelo.predict(nuevo_texto)
```

```
# Probar el modelo con un texto nuevo
nuevo_texto = ["Este producto es no es bueno, lo odio"]
prediccion = modelo.predict(nuevo_texto)

print(f"Predicción: {prediccion[0]}")
```

Resultado esperado:

Predicción: negativo

```
(venv) PS C:\Users\YaelS\Documents\GitHub\IA_Practicas\E_U1_IA>
lasificadorTxBsc/Clasificador.py
Predicción: negativo
```

CONCLUSIÓN

La práctica demuestra cómo desarrollar un clasificador de texto básico utilizando scikit-learn. A través del pipeline creado, es posible procesar texto, extraer características relevantes y realizar predicciones rápidas y precisas. Este enfoque puede ser extendido a aplicaciones más complejas, como análisis de sentimientos, filtrado de spam y categorización de noticias. Sin embargo, para mejorar el rendimiento, sería recomendable trabajar con un dataset más extenso y diverso, así como explorar técnicas avanzadas de preprocesamiento y modelado.