Institución: Tecnológico de Culiacán.

Carrera: Ingeniería en sistemas computaciones.

Materia: Inteligencia Artificial 11:00 – 12:00.

Maestro: Zuriel Dathan Mora Felix.

Tema: Tarea 3

Integrantes:

Samano Reyes Jordan Nayar

Cervantes Araujo Carlos Iván.

índice

Introducción

En este documento se hará una documentación sobre un código el cual implementa un clasificador de spam utilizando el algoritmo Naive Bayes. Procesa textos (emails), calcula probabilidades condicionales y genera un reporte de rendimiento en español.

Librerias

Lo primero que vemos en el código son las dependencias implementadas en el programa:

```
CorreoSpam.py > BayesSpam > entrenar
1   import pandas as pd
2   import numpy as np
3   from collections import defaultdict
4   import re
5   from nltk.tokenize import word_tokenize
6   from nltk.corpus import stopwords
7   import nltk
8   from sklearn.model_selection import train_test_split
9   from sklearn.metrics import accuracy_score, classification_report
```

Preprocesamiento

A continuación se crea la clase Preprocesamiento, esto con el objetivo de:

Inicializar el objeto cargando las stopwords en inglés desde NLTK y procesarlos con los siguientes pasos:

1. **Normalización**: Convierte texto a minúsculas.
2. **Limpieza**: Elimina puntuación usando regex [^\w\s].
3. **Tokenización**: Divide el texto en palabras con word_tokenize.
4. **Filtrado**: Remueve stopwords y palabras con ≤2 caracteres.

```
class Preprocesador:
    def __init__(self):
        self.stopwords = set(stopwords.words('english'))  # Carga stopwords de NLTK
    def preprocesar(self, texto):
        # Verificar que el input sea texto válido
        if not isinstance(texto, str):
            return []  # Devuelve lista vacía si no es string
        # Normalización: convertir todo a minúsculas
        texto = texto.lower()
        # Eliminar signos de puntuación (todo excepto caracteres de palabra y espacios)
        texto = re.sub(r'[^\w\s]', '', texto)
        #Tokenización: dividir el texto en palabras individuales
        tokens = word_tokenize(texto)
        # Eliminar stopwords (palabras comunes sin significado) y palabras muy cortas (longitud <= 2 caracteres)
        tokens = [palabra for palabra in tokens
                  if palabra not in self.stopwords and len(palabra) > 2]
        return tokens  # Devuelve lista de tokens limpios
```

Teorema de Bayes

Propósito:

Implementa el clasificador de spam basado en el Teorema de Bayes.

Atributos:

- P_spam: Probabilidad a priori de que un mensaje sea spam.
- P_ham: Probabilidad a priori de no spam.
- P_palabra_spam: Diccionario con P (palabra | spam).
- P_palabra_ham: Diccionario con P (palabra | ham).

Métodos:

- **entrenar (correos, etiquetas)**:
    - Calcula probabilidades a priori y condicionales con suavizado de Laplace.
    - **Ecuación de Suavizado**:

P(palabra|clase) = (frec_palabra_en_clase + 1) / (total_palabras_en_clase + tamaño_vocabulario)

```
# Calcular probabilidades condicionales con suavizado Laplace
for palabra in self.vocabulario:
    # Probabilidad de que la palabra aparezca en un mensaje SPAM (P(palabra|spam))
    self.P_palabra_spam[palabra] = (frec_spam.get(palabra, 0) + 1) / (total_palabras_spam + len(self.vocabulario))
    # Probabilidad de que la palabra aparezca en un mensaje NO SPAM (P(palabra|ham))
    self.P_palabra_ham[palabra] = (frec_ham.get(palabra, 0) + 1) / (total_palabras_ham + len(self.vocabulario))
```

- **predecir(correo)**:
    - Utiliza log-probabilidades para evitar underflow numérico.
    - Decide la clase comparando log(P_spam) + Σ log(P(palabra|spam)) vs log(P_ham) + Σ log(P(palabra|ham)).

Función “reporteF”

Propósito:

Muestra un reporte de clasificación en formato tabular en español, incluyendo:

- Precisión, Sensibilidad (Recall), F1-Score y Soporte para cada clase.
- Métricas globales: Exactitud (Accuracy) y promedios.

Ejemplo de Salida:

```
==================================================

                CLASIFICACIÓN

==================================================

                   Precisión Sensibilidad F1-score Soporte

--------------------------------------------------

NO SPAM              0.98      0.99        0.98      946

SPAM                 0.96      0.89        0.92      159

--------------------------------------------------

Promedio macro       0.97      0.94        0.95      1105

Promedio ponderado   0.97      0.97        0.97      1105

==================================================

Exactitud global del modelo: 0.9745

==================================================
```

Flujo Principal

1. **Carga de Datos**:
    - Lee el archivo Spamrial.csv y renombra columnas a label (1=spam, 0=ham) y text.
2. **Preprocesamiento**:
    - Aplica la clase Preprocesador a todos los textos.
3. **División de Datos**:
    - Separa en 80% entrenamiento y 20% prueba usando train_test_split.
4. **Entrenamiento**:
    - Crea una instancia de BayesSpam y entrena con los datos de entrenamiento.
5. **Evaluación**:
    - Predice las etiquetas de prueba y genera el reporte en español.

Instrucciones de Uso

1. Ejecutar el script en un entorno con las dependencias instaladas.
2. Asegurarse de que el archivo Spamrial.csv esté en el directorio correcto.
3. Los resultados se imprimirán en consola con el formato especificado.