Análisis de Sentimientos con Python

Amado Moncada Castro, Edwin Farid Bolaño, Jesus Eduardo Cañas

17 de diciembre de 2024

1. Introducción

Este documento explica el código que realiza un análisis de sentimientos de reseñas de productos utilizando Python. Se utiliza un modelo de **regresión logística** y el enfoque de **TF-IDF** (Term Frequency-Inverse Document Frequency) para convertir las reseñas en vectores numéricos que pueden ser procesados por el modelo.

2. Importación de Librerías

El código comienza importando las siguientes librerías necesarias para el procesamiento de datos y el análisis de sentimientos:

Listing 1: Importación de librerías

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
```

- **pandas**: Se utiliza para manipular los datos y almacenarlos en un DataFrame.
- **train_test_split**: Permite dividir los datos en un conjunto de entrenamiento y uno de prueba.
- **TfidfVectorizer**: Convierte el texto en vectores numéricos utilizando el método TF-IDF.
- **LogisticRegression**: Modelo utilizado para clasificar las reseñas como positivas o negativas.

- **accuracy_score** y **classification_report**: Permiten evaluar el desempeño del modelo.

3. Creación de Datos de Entrada

Se define un conjunto de datos con reseñas de productos y sus sentimientos asociados. Las reseñas se etiquetan como 1 para positivas y 0 para negativas.

Listing 2: Definición de datos de entrada

```
data = {
    'review': [
        'Este producto es increble, lo recomiendo mucho',
        'Muy mala calidad, no lo volver a comprar',
        'Excelente relacin calidad-precio, muy satisfecho',
        # Ms reseas...
    ],
    'sentiment': [
        1, 0, 1,  # 1 = Positiva, 0 = Negativa
        # Ms etiquetas de sentimiento...
    ]
}
```

4. Creación del DataFrame

El siguiente paso es convertir los datos de reseñas y sentimientos en un *DataFrame* de pandas para facilitar su manipulación:

Listing 3: Conversión a DataFrame

```
df = pd.DataFrame(data)
```

5. División de Datos en Conjuntos de Entrenamiento y Prueba

Luego, se dividen los datos en conjuntos de entrenamiento (70 %) y prueba (30 %) usando la función *train_test_split*.

Listing 4: División de los datos

```
X_train, X_test, y_train, y_test = train_test_split(
    df['review'], df['sentiment'], test_size=0.3, random_state=42)
```

6. Definición de Stopwords

Las *stopwords* son palabras comunes que se eliminan durante el procesamiento del texto. Se define una lista de stopwords en español:

Listing 5: Stopwords en español

```
stopwords_espanol = [
    'de', 'la', 'que', 'el', 'en', 'y', 'a', 'los', 'se', 'del', 'las'
    # M s stopwords...
]
```

7. Vectorización de Texto con TF-IDF

El siguiente paso es convertir las reseñas en vectores numéricos usando el enfoque TF-IDF. Se utiliza la clase `TfidfVectorizer` para transformar las reseñas.

Listing 6: Vectorización TF-IDF

```
tfidf = TfidfVectorizer(stop_words=stopwords_espanol, lowercase=True)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

Aquí también se agregan algunas stopwords adicionales específicas, como *pesimo*, *malo*, etc.

8. Entrenamiento del Modelo de Regresión Logística

El modelo de regresión logística se entrena con los datos vectorizados.

Listing 7: Entrenamiento del modelo

```
model = LogisticRegression()
model.fit(X_train_tfidf, y_train)
```

9. Evaluación del Modelo

Después de entrenar el modelo, se hacen predicciones sobre el conjunto de prueba y se evalúa la precisión del modelo.

Listing 8: Evaluación del modelo

```
y_pred = model.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
print(classification_report(y_test, y_pred))
```

10. Predicción de Sentimiento de Nuevas Reseñas

Se define una función para predecir el sentimiento de nuevas reseñas que ingrese el usuario:

Listing 9: Predicción de Sentimiento

```
def predecir_sentimiento(rese a):
    rese a_tfidf = tfidf.transform([rese a])
    prediccion = model.predict(rese a_tfidf)
    if prediccion[0] == 1:
        return "Positiva"
    else:
        return "Negativa"
```

11. Interacción con el Usuario

Finalmente, el código permite al usuario ingresar una reseña y obtener una predicción sobre su sentimiento:

Listing 10: Interacción con el usuario

```
rese a_usuario = input("Ingresa tu rese a de producto para analizar
sentimiento = predecir_sentimiento(rese a_usuario)
print(f"La rese a es clasificada como: {sentimiento}")
```

12. Desafios

° Mejorar la comprensión y ejecución en Python. Este problema se solventó estudiando los conceptos básicos del lenguaje.

° Precisión del modelo. Se solventó agregando una línea que contiene palabras que no son relevantes para el entrenamiento del modelo, junto a esto, se agregaron más datos de entrenamiento.

° Falta de conocimiento sobre el uso de las librerías de forma óptima. Al leer la documentación de las librerías, se solventó este problema.

13. Conclusión

Este código entrena un modelo de regresión logística para clasificar reseñas de productos como positivas o negativas. Utiliza la vectorización TF-IDF para convertir el texto en vectores numéricos y realizar predicciones sobre nuevos datos. Además, permite la interacción con el usuario para predecir el sentimiento de nuevas reseñas de productos.