

Caso de análisis:

La gerencia general ha aprobado el proyecto de la célula Data Science para el análisis de sentimientos, por lo que ahora tiene que iniciar el preprocesamiento de texto. Todo el equipo está muy motivado en poner manos a la obra, ya que es la oportunidad de poner en práctica lo aprendido. Las primeras tareas que se van a realizar son: tokenización, limpieza y reducción del vocabulario.


✓ Consignas

1. Importación de librerías. Se realizará una segunda parte del análisis, a la cual vamos a llamar "Preprocesamiento de datos". Primero vamos a importar todas las librerías que vamos a usar en el análisis, las que consideremos necesarias.
2. Limpieza, tokenización y reducción del vocabulario. a. Se definirá una función similar a la vista en los ejemplos, pero debemos personalizarla, en otras palabras, no copiar los ejemplos textualmente, solo basarnos en ellos. En la misma función, podemos realizar la tokenización y limpieza, o se pueden crear dos funciones por separado; lo importante es que especifiquemos dónde aplicamos la tokenización y la realicemos con RegexpTokenizer. Para ello, se debe usar como parámetro el patrón de expresión regular que se diseñó en la consigna 4 de la actividad anterior. b. Se codificará una función para lematización y otra para stemming, que lea el corpus de los comentarios, y entregue el corpus reducido a lemas y raíces respectivamente. Debemos basarnos en los ejemplos, pero no debemos copiarlos textualmente.
3. Split de la muestra. Realizaremos la división de la muestra en entrenamiento y test, utilizando el método train_test_split de la librería sklearn. Se destinará el 20 % de muestra para el test y el 80 % para el entrenamiento.
4. Distribución del target según muestra, train y test. Realizaremos un diagrama de barras de los porcentajes del target (bueno, malo, o info) según muestra train y test. Las distribuciones pueden no salir tan similares, debido a que la muestra test tiene baja materialidad. Lo importante es que por lo menos se mantenga el orden.

1) Importación de librerías

```
import pandas as pd
import numpy as np
import re
import nltk
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer, PorterStemmer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns

nltk.download('wordnet')
nltk.download('stopwords')
```

 [nltk_data] Downloading package wordnet to /root/nltk_data...
 [nltk_data] Downloading package stopwords to /root/nltk_data...
 [nltk_data] Unzipping corpora/stopwords.zip.
 True

2) limpieza y tokenización

```
import re

def limpiar_tokenizar(texto):
    texto = texto.lower()
    tokenizer = RegexpTokenizer(r'^\s^\.^;]+')
    tokens = tokenizer.tokenize(texto)
    return tokens
texto = 'vi viene última . fff , ♥ 🍕 :)'
tokens = limpiar_tokenizar(texto)
print(tokens)
```

b. lematización y stemming

```

from nltk.stem import WordNetLemmatizer, PorterStemmer

# función para lematizar
def lematizar(tokens):
    lemmatizer = WordNetLemmatizer()
    lemas = [lemmatizer.lemmatize(token) for token in tokens]
    return lemas

# función para stemming
def stemming(tokens):
    stemmer = PorterStemmer()
    raices = [stemmer.stem(token) for token in tokens]
    return raices

```

ejemplos:

```

data = {
    'texto': [
        "Este es un buen ejemplo.",
        "Este es un mal ejemplo.",
        "Este es un ejemplo informativo.",
        "comentario adicional",
        "comentario malo",
        "comentario bueno"
    ],
    'target': ['bueno', 'malo', 'info', 'adicional', 'malo', 'bueno']
}
df = pd.DataFrame(data)
df['tokens'] = df['texto'].apply(limpiar_tokenizar)
df['lemas'] = df['tokens'].apply(lematizar)
df['raices'] = df['tokens'].apply(stemming)

print(df[['lemas', 'texto', 'tokens', 'raices']])

```

	lemas	texto \
0	[este, e, un, buen, ejemplo]	Este es un buen ejemplo.
1	[este, e, un, mal, ejemplo]	Este es un mal ejemplo.
2	[este, e, un, ejemplo, informativo]	Este es un ejemplo informativo.
3	[comentario, adicional]	comentario adicional
4	[comentario, malo]	comentario malo
5	[comentario, bueno]	comentario bueno

	tokens	raices
0	[este, es, un, buen, ejemplo]	[est, es, un, buen, ejemplo]
1	[este, es, un, mal, ejemplo]	[est, es, un, mal, ejemplo]
2	[este, es, un, ejemplo, informativo]	[est, es, un, ejemplo, informativo]
3	[comentario, adicional]	[comentario, adicicion]
4	[comentario, malo]	[comentario, malo]
5	[comentario, bueno]	[comentario, bueno]

3) split de la muestra

```

train, test = train_test_split(df, test_size=0.2, random_state=42)
print("Entrenamiento:")
print(train)
print("Prueba:")
print(test)

```

```

Entrenamiento:

```

	texto	target \
5	comentario bueno	bueno
2	Este es un ejemplo informativo.	info
4	comentario malo	malo
3	comentario adicional	adicional

	tokens	lemas \
5	[comentario, bueno]	[comentario, bueno]
2	[este, es, un, ejemplo, informativo]	[este, e, un, ejemplo, informativo]
4	[comentario, malo]	[comentario, malo]
3	[comentario, adicional]	[comentario, adicional]

	raices
5	[comentario, bueno]
2	[est, es, un, ejemplo, informativo]
4	[comentario, malo]
3	[comentario, adicicion]

Prueba:

	texto	target	tokens \
0	Este es un buen ejemplo.	bueno	[este, es, un, buen, ejemplo]
1	Este es un mal ejemplo.	malo	[este, es, un, mal, ejemplo]

	lemas	raices
0	[este, e, un, buen, ejemplo]	[est, es, un, buen, ejemplo]

```
1 [este, e, un, mal, ejemplo] [est, es, un, mal, ejemplo]
```

4) target

```
import matplotlib.pyplot as plt
import seaborn as sns

train['set'] = 'train'
test['set'] = 'test'
df_all = pd.concat([train, test])

# gráfica
plt.figure(figsize=(10, 6))
sns.countplot(x='target', hue='set', data=df_all)
plt.title('Distribución del target según muestra train y test')
plt.xlabel('Target')
plt.ylabel('Cantidad')
plt.legend(title='Set')
plt.show()
```

