Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Integrantes:

José Jesús Ramírez Cruz 202052478

Gabriel Reyes Leal 202053516

Alfonso Saldaña Campos 202056307

Periodo: Primavera 2024

Materia: Minería de Datos

Manual Técnico

Nombre del Docente: María Beatriz Bernabé Loranca

Manual Técnico para el Análisis de Sentimientos

DESCRIPCIÓN GENERAL

Este manual técnico proporciona una guía detallada sobre cómo ejecutar un análisis de sentimientos en tweets utilizando Python. El análisis incluye el preprocesamiento del texto, el cálculo de etiquetas de sentimiento basado en un diccionario, la visualización de los resultados y el entrenamiento de un modelo Naive Bayes para la clasificación de sentimientos.

REQUISITOS PREVIOS

Antes de ejecutar el script, asegúrese de tener las siguientes dependencias instaladas:

```
pip install re ssl openpyxl nltk scikit-learn joblib matplotlib numpy unidecode
```

Además, necesita descargar los recursos de NLTK:

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

ESTRUCTURA DEL CÓDIGO

1. Importación de Librerías

El primer paso en la creación de nuestro script de análisis de sentimientos es importar las librerías necesarias. Este conjunto de librerías nos permitirá realizar diversas tareas, desde la manipulación de datos y el procesamiento de texto hasta la visualización y el machine learning. Entre las librerías importadas, re se utiliza para manejar expresiones regulares, lo cual es fundamental para limpiar y preparar el texto. ssl nos ayudará a gestionar problemas potenciales relacionados con certificados SSL, mientras que openpyxl nos permitirá trabajar con archivos de Excel, que es el formato en el que tenemos almacenados nuestros tweets. Además, se utilizan nltk para el procesamiento del lenguaje natural, scikit-learn para construir y evaluar modelos de machine learning, joblib para guardar y cargar modelos, y matplotlib junto con numpy para la visualización de datos y operaciones numéricas. Finalmente,

unidecode nos ayuda a normalizar el texto, eliminando acentos y caracteres especiales.

```
import re
import ssl
import openpyxl
from nltk import WordNetLemmatizer
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
import joblib
import nltk
import matplotlib.pyplot as plt
import numpy as np
import csv
from collections import Counter
from unidecode import unidecode
```

2. Configuración de SSL

En algunos entornos, especialmente aquellos con restricciones de seguridad o configuraciones de red específicas, puede haber problemas al intentar descargar recursos externos debido a la verificación de certificados SSL. Para mitigar esto, se incluye una línea de código que deshabilita la verificación de certificados SSL. Esta configuración permite que las funciones de descarga de NLTK operen sin interrupciones, asegurando que todas las librerías y datos necesarios para el procesamiento del lenguaje natural estén disponibles. Aunque esta práctica puede ser útil en entornos de desarrollo o en situaciones específicas, es importante ser cautelosos con su uso en producción debido a las implicaciones de seguridad que conlleva desactivar la verificación SSL

```
ssl._create_default_https_context = ssl._create_unverified_context
```

3. Definición del Diccionario de Sentimientos

El análisis de sentimientos se basa en un diccionario que contiene listas de palabras clasificadas según su connotación emocional. En este caso, se definen dos listas: una para palabras con connotaciones muy positivas (palabras_muy_bueno) y otra para palabras con connotaciones muy negativas (palabra_muy_mal). Estas listas se utilizarán para evaluar el contenido emocional de los tweets. El diccionario principal agrupa estas listas, permitiendo un acceso fácil y estructurado durante el análisis. Las palabras seleccionadas para estas listas reflejan términos comunes en el lenguaje cotidiano que indican sentimientos fuertes, y se utilizarán para contar ocurrencias y determinar el sentimiento general de los textos analizados.

```
palabras_muy_bueno = ["primera", "futura", "bien", ...]
palabra_muy_mal = ["mala persona", "mala", "pena", ...]
```

```
diccionario_principal = {"palabras_muy_bueno": palabras_muy_bueno, "palabra_muy_mal": palabra_muy_mal}
```

4. Definición de Stop Words

Las stop words son palabras comunes que, en general, no aportan información significativa al análisis de texto y, por lo tanto, se excluyen durante el preprocesamiento. En este script, se define un conjunto de stop words específico para el idioma español, que incluye artículos, preposiciones, pronombres y otras palabras frecuentes. Al eliminar estas palabras, se reduce el ruido en los datos y se mejora la precisión del análisis de sentimientos. La elección de estas stop words está basada en las características del idioma español y está diseñada para garantizar que las palabras clave que realmente aportan información sobre el sentimiento se mantengan en el texto procesado.

```
stop_words = {
    'la', 'el', 'y', 'yo', 'ella', 'es', 'a', ...
}
```

5. Funciones de Preprocesamiento de Texto

El preprocesamiento del texto es una etapa crucial que prepara los datos para el análisis posterior. La función preprocess_text realiza varias operaciones: elimina enlaces web, menciones de usuarios, y caracteres no alfabéticos; convierte el texto a minúsculas; elimina stop words; y aplica lematización, que es el proceso de reducir las palabras a su forma base o raíz. Estas operaciones transforman el texto en una versión más limpia y consistente, que es más fácil de analizar y menos propensa a errores de interpretación. La lematización, en particular, ayuda a tratar palabras diferentes con significados similares de manera uniforme, mejorando la calidad del análisis de sentimientos.

```
def preprocess_text(text):
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'@[A-Za-z0-9_]+', '', text)
    text = re.sub(r'[^A-Za-zñÑáéíóúÁÉÍÓÚ\s]', ' ', text)
    words = text.lower().split()
    words = [word for word in words if word not in stopwords.words('spanish')]
    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(word) for word in words]
    return ' '.join(words)
```

6. Función para Etiquetar Sentimientos

La función get_sentiment_labels se encarga de asignar etiquetas de sentimiento (bueno, malo, neutral) a un texto basado en la presencia de palabras positivas y negativas definidas en el diccionario de sentimientos. La función cuenta las ocurrencias de palabras de ambas categorías y compara estos conteos. Si las palabras positivas predominan, el texto se etiqueta como "bueno"; si las palabras negativas son más numerosas, se etiqueta como "malo"; y si hay un equilibrio, se etiqueta como "neutral". Este enfoque basado en conteo es simple pero efectivo para proporcionar una indicación inicial del sentimiento en textos cortos como los

tweets.

```
def get_sentiment_labels(text):
    count_bueno = sum(word in diccionario_principal["palabras_muy_bueno"] for word
    count_mal = sum(word in diccionario_principal["palabra_muy_mal"] for word in te
    if count_bueno > count_mal:
        return "bueno"
    elif count_mal > count_bueno:
        return "malo"
    else:
        return "neutral"
```

7. Lectura de Tweets desde un Archivo Excel

La función read_tweets_from_excel permite leer y extraer tweets almacenados en un archivo de Excel. Utilizando la librería openpyxl, la función abre el archivo especificado, accede a la hoja de trabajo activa y recoge los valores de una columna específica que contiene los tweets. Este método es flexible y permite gestionar grandes volúmenes de datos de manera eficiente. La capacidad de trabajar con archivos de Excel es particularmente útil en situaciones donde los datos de los tweets se han recopilado y almacenado en este formato, facilitando así su análisis y procesamiento.

```
def read_tweets_from_excel(file_path):
    wb = openpyxl.load_workbook(file_path)
    sheet = wb.active
    tweets = [cell.value for cell in sheet['A']]
    return tweets
```

8. Procesamiento de Tweets y Obtención de Sentimientos

La función process_tweets_and_get_sentiments es el núcleo del flujo de trabajo de análisis de sentimientos. Esta función toma una lista de tweets, los preprocesa utilizando la función preprocess_text, y luego aplica la función get_sentiment_labels para etiquetar cada tweet con un sentimiento. Este proceso en dos pasos garantiza que cada tweet esté limpio y en un formato adecuado antes de la evaluación de sentimientos. Al final, se obtiene una lista de etiquetas de sentimientos correspondientes a los tweets, lo cual facilita el análisis cuantitativo y la visualización de los resultados.

```
def process_tweets_and_get_sentiments(tweets):
    preprocessed_tweets = [preprocess_text(str(tweet)) for tweet in tweets]
    sentiment_labels = [get_sentiment_labels(words) for words in preprocessed_tweets]
    return sentiment_labels
```

9. Cálculo de Porcentajes de Sentimientos

Para obtener una visión general de los sentimientos expresados en los tweets, la función calculate_sentiment_percentages calcula los porcentajes de tweets positivos, negativos y neutrales. La función toma las listas de etiquetas de sentimiento generadas previamente y cuenta las ocurrencias de cada tipo de sentimiento. Luego, calcula los porcentajes relativos al número total de tweets para cada conjunto de datos. Este análisis cuantitativo permite comparar de manera efectiva la distribución de sentimientos entre diferentes grupos de tweets, proporcionando una visión clara de las tendencias emocionales en los datos.

```
def calculate_sentiment_percentages(sentiment_labels1, sentiment_labels2):
    total_tweets1 = len(sentiment_labels1)
    total_tweets2 = len(sentiment_labels2)
    unique_labels1, counts1 = np.unique(sentiment_labels1, return_counts=True)
    unique_labels2, counts2 = np.unique(sentiment_labels2, return_counts=True)
    counts_dict1 = dict(zip(unique_labels1, counts1))
    counts_dict2 = dict(zip(unique_labels2, counts2))
    positive_tweets1 = counts_dict1.get("bueno", 0)
    negative_tweets1 = counts_dict1.get("malo", 0)
    neutral_tweets1 = total_tweets1 - positive_tweets1 - negative_tweets1
    positive_tweets2 = counts_dict2.get("bueno", 0)
    negative_tweets2 = counts_dict2.get("malo", 0)
    neutral_tweets2 = total_tweets2 - positive_tweets2 - negative_tweets2
    positive_percent1 = (positive_tweets1 / total_tweets1) * 100
    negative_percent1 = (negative_tweets1 / total_tweets1) * 100
    neutral_percent1 = (neutral_tweets1 / total_tweets1) * 100
    positive_percent2 = (positive_tweets2 / total_tweets2) * 100
    negative_percent2 = (negative_tweets2 / total_tweets2) * 100
    neutral_percent2 = (neutral_tweets2 / total_tweets2) * 100
    return (
        positive_percent1, negative_percent1, neutral_percent1,
        positive_percent2, negative_percent2, neutral_percent2,
        total_tweets1, total_tweets2,
        positive_tweets1, positive_tweets2,
        negative_tweets1, negative_tweets2,
        neutral_tweets1, neutral_tweets2
    )
```

10. Visualización de la Comparación de Sentimientos

La función plot_comparison genera una visualización gráfica que compara los sentimientos entre dos conjuntos de tweets. Utilizando matplotlib, la función crea un gráfico de barras que muestra los porcentajes de tweets positivos, negativos y neutrales para cada grupo. Además, se incluye una tabla que resume los datos numéricos, proporcionando una representación clara y concisa de los resultados. Esta visualización es una herramienta poderosa para comunicar los hallazgos de manera intuitiva, facilitando la comprensión de las diferencias y similitudes en los sentimientos expresados en los tweets analizados.

```
def plot_comparison(positive_percent1, negative_percent1, neutral_percent1, positive
                    neutral_percent2, total_tweets1, total_tweets2, positive_tweets1
                    negative_tweets1, negative_tweets2, neutral_tweets1, neutral_twe
    labels = ['Positivos', 'Negativos', 'Neutrales']
    values_persona1 = [positive_percent1, negative_percent1, neutral_percent1]
    values_persona2 = [positive_percent2, negative_percent2, neutral_percent2]
    x = range(len(labels))
    plt.figure(figsize=(12, 6))
    plt.bar(x, values_persona1, width=0.3, label='Claudia', color='blue', align='cen
    plt.bar(x, values_persona2, width=0.3, label='Xochitl', color='orange', align='e
    plt.xticks(x, labels)
    plt.ylabel('Porcentaje')
    plt.title('Comparación de sentimientos')
    plt.legend()
    table_data = [
        ['Total Tweets', total_tweets1, total_tweets2],
        ['Positivos', positive_tweets1, positive_tweets2],
        ['Negativos', negative_tweets1, negative_tweets2],
        ['Neutrales', neutral_tweets1, neutral_tweets2]
    ]
    table = plt.table(cellText=table_data, colLabels=['', 'Claudia', 'Xochitl'], loc
    table.auto_set_font_size(False)
    table.set_fontsize(10)
    table.scale(1, 1.5)
    plt.subplots_adjust(right=0.7)
    plt.show()
```

11. Entrenamiento del Modelo Naive Bayes

Para llevar el análisis de sentimientos un paso más allá, la función train_naive_bayes_model entrena un modelo de clasificación Naive Bayes. Este modelo utiliza un CountVectorizer para transformar el texto en una representación numérica basada en la frecuencia de palabras, y luego entrena el modelo con los datos etiquetados. Una vez entrenado, el modelo y el vectorizador se guardan en archivos utilizando joblib, permitiendo su reutilización sin necesidad de reentrenamiento. Este enfoque basado en machine learning puede proporcionar

un análisis de sentimientos más preciso y flexible, capaz de generalizar a nuevos datos no vistos durante el entrenamiento.

```
def train_naive_bayes_model(X, y):
    vectorizer = CountVectorizer()
    X_str = [str(doc) for doc in X]
    X_vectorized = vectorizer.fit_transform(X_str)
    model = MultinomialNB()
    model.fit(X_vectorized, y)
    joblib.dump(model, 'naive_bayes_model.pkl')
    joblib.dump(vectorizer, 'count_vectorizer.pkl')
    return model, vectorizer
```

12. Funciones Auxiliares para la Limpieza y Extracción de Texto

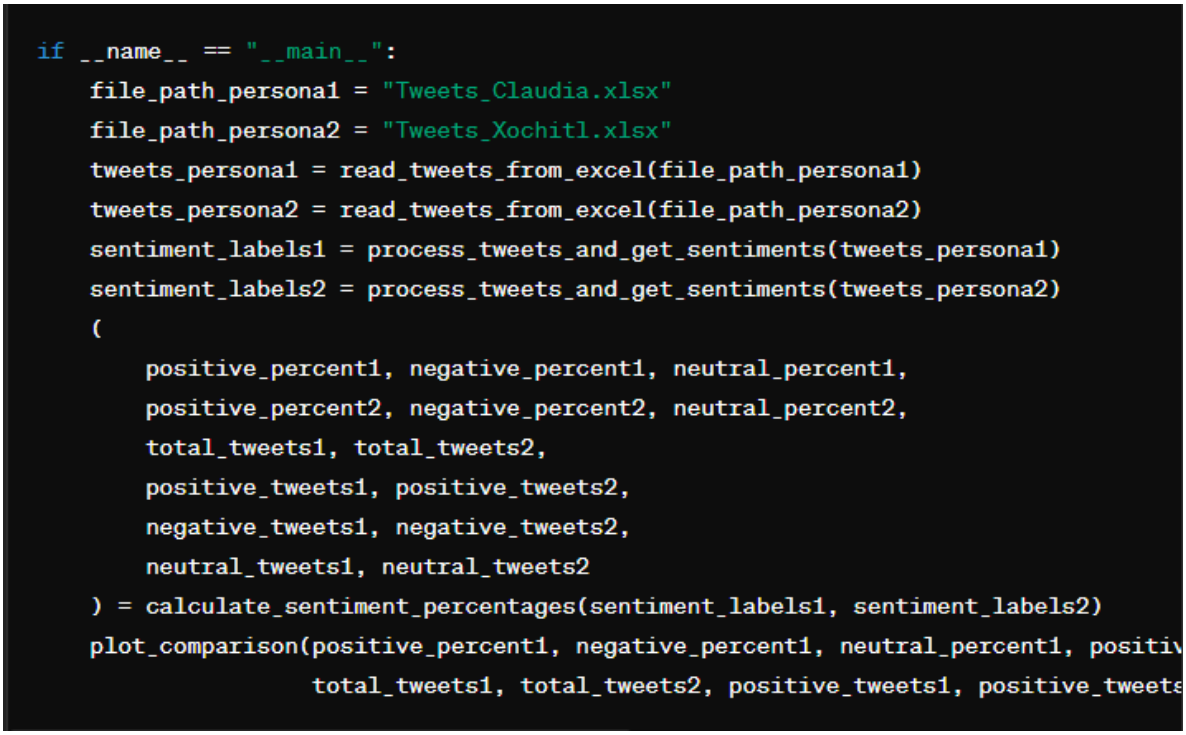
Las funciones auxiliares clean_text y extract_words_from_excel proporcionan herramientas adicionales para la limpieza y preparación de datos. clean_text realiza una limpieza básica del texto, eliminando URLs, menciones de usuarios y caracteres no alfabéticos, y convirtiendo el texto a minúsculas. extract_words_from_excel lee un archivo de Excel, limpia el texto y extrae las palabras, preparándolas para su uso en el análisis de sentimientos o en el entrenamiento de modelos. Estas funciones complementarias aseguran que los datos estén en el formato adecuado para el procesamiento posterior, mejorando la calidad y consistencia del análisis.

```
def clean_text(text):
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r"@\w+", "", text)
    text = re.sub(r"[^A-Za-z\s]", "", text)
    return text.lower()

def extract_words_from_excel(file_path):
    workbook = openpyxl.load_workbook(file_path)
    sheet = workbook.active
    words = []
    for row in sheet.iter_rows(values_only=True):
        for cell in row:
            if cell:
                words.extend(clean_text(cell).split())
    return words
```

13. Main Function

La función principal main ejecuta el flujo completo del análisis de sentimientos, desde la lectura de los tweets hasta la visualización de los resultados. Primero, lee los tweets de dos archivos de Excel diferentes, preprocesa los textos y obtiene las etiquetas de sentimiento para cada conjunto de tweets. Luego, calcula los porcentajes de sentimientos y genera una visualización comparativa. Este flujo integrado permite realizar un análisis completo y detallado de los sentimientos expresados en los tweets de manera eficiente. La estructura modular del código facilita su modificación y expansión para adaptarse a diferentes necesidades y conjuntos de datos.



```
if __name__ == "__main__":
    file_path_persona1 = "Tweets_Claudia.xlsx"
    file_path_persona2 = "Tweets_Xochitl.xlsx"
    tweets_persona1 = read_tweets_from_excel(file_path_persona1)
    tweets_persona2 = read_tweets_from_excel(file_path_persona2)
    sentiment_labels1 = process_tweets_and_get_sentiments(tweets_persona1)
    sentiment_labels2 = process_tweets_and_get_sentiments(tweets_persona2)
    (
        positive_percent1, negative_percent1, neutral_percent1,
        positive_percent2, negative_percent2, neutral_percent2,
        total_tweets1, total_tweets2,
        positive_tweets1, positive_tweets2,
        negative_tweets1, negative_tweets2,
        neutral_tweets1, neutral_tweets2
    ) = calculate_sentiment_percentages(sentiment_labels1, sentiment_labels2)
    plot_comparison(positive_percent1, negative_percent1, neutral_percent1, positiv
                    total_tweets1, total_tweets2, positive_tweets1, positive_tweets
```