



UNIVERSIDAD DEL BÍO-BÍO

# Análisis de sentimiento en base a texto.

Desarrolladores: Roberto Arriagada

Bastián Araya

# Metadata

Nr	Descripción	-----
C1	Current code version	V1.5
C2	Permanent link to code/repository used for this code version	Google colab
C3	Permanent link to reproducible capsule	
C4	Legal code license	
C5	Code versioning system used	Google colab
C6	Software code languages, tools and services used	python
C7	Compilation requirements, operating environments and dependencies	
C8	If available, link to developer documentation/manual	
C9	Support email for questions	

# 1. Motivación

Inicialmente, el inicio de este proyecto estaba centrado en el análisis de emojis. Sin embargo, tras enfrentarnos a la falta de un dataset apropiado para este propósito, se tomó la decisión de ajustar la dirección del proyecto. Durante la implementación inicial con emojis, se encontraron obstáculos significativos, evidenciados por valores negativos y desafíos asociados al dataset que presentaba tres opciones de sentimiento (positivo, neutral y negativo), lo cual afectaba la calidad de los resultados y dificulta la obtención de conclusiones significativas.

Ante estas dificultades, se ha adaptado la estrategia del proyecto para enfocarse exclusivamente en el análisis de sentimientos basados en texto de tweets. Este cambio de enfoque permitirá superar las limitaciones previas y brindará una base más sólida para obtener conclusiones más robustas. La atención se centrará en la comprensión de la carga emocional y la intención detrás de los mensajes en Twitter, proporcionando así una perspectiva valiosa sobre el análisis de sentimientos en el ámbito de las redes sociales.

La ejecución de un proyecto de análisis de sentimiento basado en texto mediante técnicas de deep learning es crucial en el panorama actual, donde la vasta cantidad de datos generados por usuarios en plataformas digitales y redes sociales requiere una comprensión sofisticada de las emociones subyacentes.

Este enfoque no solo aspira a transformar el vasto océano de opiniones y comentarios en información estructurada y valiosa, sino que también busca dotar a las empresas con herramientas avanzadas para la toma de decisiones informada, la personalización de servicios, la mejora continua de la satisfacción del cliente y la identificación proactiva de tendencias en el mercado.

## 2. Descripción de Software

### 2.1. Arquitectura de Software

#### 1. Adquisición de datos:

- Recopilación de datos etiquetados para entrenar el modelo (conjunto de entrenamiento).
- Posiblemente, datos no etiquetados para el entrenamiento no supervisado o la mejora del modelo.
- Conjunto de datos de prueba para evaluar el rendimiento del modelo.

#### 2. Preprocesamiento de datos:

- Tokenización: Dividir el texto en palabras o subunidades significativas.
- Limpieza de texto: Eliminación de puntuaciones, caracteres especiales, y stop words.
- Normalización: Convertir todas las palabras a minúsculas.
- Padding o truncamiento: Asegurarse de que todas las secuencias de texto tengan la misma longitud.

#### 3. Representación de texto:

- Convertir las palabras a vectores utilizando técnicas como Word Embeddings (Word2Vec, GloVe).

#### 4. Construcción del modelo:

- Utilizar una arquitectura de red neuronal profunda. Ejemplos incluyen redes neuronales recurrentes (RNN), redes neuronales convolucionales (CNN) o modelos más avanzados como transformers.
- Añadir capas de Embedding para procesar la representación de texto.
- Capas ocultas para aprender patrones y características.
- Capa de salida con función de activación para clasificación binaria o softmax para clasificación multiclase.

#### 5. Entrenamiento del modelo:

- Definir la función de pérdida (loss function).
- Seleccionar un optimizador (SGD, Adam, RMSprop).
- Entrenar el modelo utilizando el conjunto de entrenamiento.
- Ajustar hiperparámetros, si es necesario, mediante validación cruzada.

#### 6. Evaluación del modelo:

- Evaluar el modelo utilizando el conjunto de prueba.
- Calcular métricas como precisión, recall, F1-score.
- Analizar la matriz de confusión.

#### 7. Despliegue del modelo:

- Integrar el modelo en una aplicación o servicio.
- Exponer una API para la predicción de sentimientos.

#### 8. Optimización y mantenimiento:

- Realizar ajustes y optimizaciones según sea necesario.
- Monitorizar el rendimiento del modelo en producción.
- Actualizar el modelo periódicamente con nuevos datos.

## 2.2. Funcionalidad del Software

### 1. Funcionalidad de Procesamiento de Datos:

- El programa es capaz de preprocesar y procesar texto de manera eficiente, tokenizado y limpiando datos para su posterior análisis.

### 2. Almacenamiento y Recuperación de Datos:

- El software puede trabajar con conjuntos de datos almacenados localmente o recuperar datos de fuentes externas para realizar el análisis de sentimiento.

### 3. Conectividad y Comunicación:

- Podría conectarse a fuentes de datos en línea o a bases de datos para obtener textos a analizar. También podría proporcionar salidas a través de servicios web o integrarse con otras aplicaciones.

### 4. Automatización:

- La capacidad para procesar grandes cantidades de texto de manera automática, realizando análisis de sentimiento sin intervención manual.

### 5. Seguridad:

- Implementa medidas de seguridad para proteger los modelos de análisis de sentimiento y los datos procesados, asegurando la privacidad y confidencialidad.

### 6. Compatibilidad:

- Diseñado para ser compatible con diferentes versiones de Python y bibliotecas relacionadas, permitiendo su ejecución en diversos entornos.

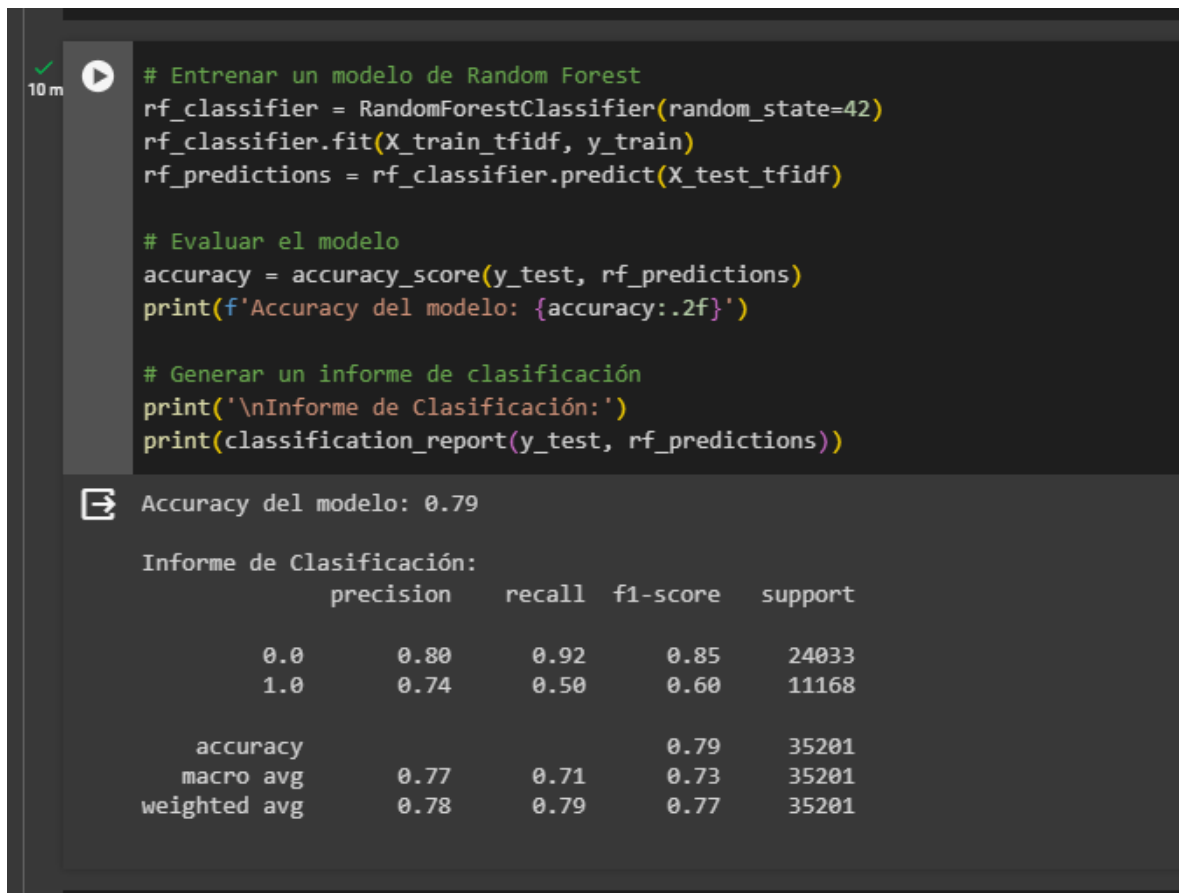
### 7. Personalización:

- Podría permitir la personalización de modelos para adaptarse a dominios específicos o vocabularios particulares mediante el ajuste de hiper parámetros o la inclusión de datos personalizados.

## 8. Documentación:

- Proporciona documentación detallada, incluyendo manuales de usuario, guías de instalación y documentación técnica que facilite la comprensión y utilización del software.

### 3. Ejemplo ilustrativo



```
# Entrenar un modelo de Random Forest
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train_tfidf, y_train)
rf_predictions = rf_classifier.predict(X_test_tfidf)

# Evaluar el modelo
accuracy = accuracy_score(y_test, rf_predictions)
print(f'Accuracy del modelo: {accuracy:.2f}')

# Generar un informe de clasificación
print('\nInforme de Clasificación:')
print(classification_report(y_test, rf_predictions))
```

Accuracy del modelo: 0.79

Informe de Clasificación:

	precision	recall	f1-score	support
0.0	0.80	0.92	0.85	24033
1.0	0.74	0.50	0.60	11168
accuracy			0.79	35201
macro avg	0.77	0.71	0.73	35201
weighted avg	0.78	0.79	0.77	35201

En la imagen anterior ver como se utilizo el codigo para utilizar randomforest con el dataset y obtener resultados de esto, donde como se puede apreciar que ha obtenido un buen puntaje para realizar análisis de sentimiento en tu conjunto de datos ya que el obtuvo un 79% de precisión, lo cual es bastante bueno en términos generales. Indica que el 79% de las predicciones realizadas por el modelo son correctas, lo cual es positivo. También el F1-score, que es una medida que combina precisión y recall, también es razonablemente alto, especialmente para la clase 0.0 (negativa) con un 85%, lo cual da para entender que el modelo es capaz de identificar correctamente la clase neutra en la mayoría de los casos y también tiene una buena proporción de predicciones correctas para esta clase, aun así se puede ver que la clase 1.0 tiene un recall bajo por lo que se puede decir que falta mejorar este aspecto del randomforest.



Ahora en la siguiente imagen se muestra la ejecución del código para una red neuronal convulsionar:

```
# Compilar el modelo
cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Entrenar el modelo CNN
cnn_model.fit(X_train_seq, y_train, epochs=5, batch_size=32, validation_split=0.2)
```

```
*** Epoch 1/5
3521/3521 [=====] - 2784s 790ms/step - loss: 0.4403 - accuracy: 0.8012 - val_loss: 0.4089 - val_accuracy: 0.8140
Epoch 2/5
3521/3521 [=====] - 2832s 804ms/step - loss: 0.3789 - accuracy: 0.8343 - val_loss: 0.4106 - val_accuracy: 0.8145
Epoch 3/5
9/3521 [.....] - ETA: 46:47 - loss: 0.3574 - accuracy: 0.8438
```

Y se puede concluir que la función de pérdida disminuye desde el primer al segundo epoch, lo cual es un buen indicador ya que el modelo está mejorando en la tarea y también la precisión aumenta, lo cual es otra señal positiva de mejora. Por falta de tiempo cada epoch demoraba alrededor de 1 hora por lo que se suspendió y se tomó esos dos epoch por lo que se puede deducir que el accuracy completo debería ser mayor a 0.8 lo que significa que aumenta de buena forma la precisión de esta.

## 4. Impacto

### Nuevas Preguntas de Investigación:

La implementación del código ha generado una nueva interrogante en torno al análisis de sentimientos basado en texto, específicamente en el contexto de las redes sociales. Se observa que el dataset ha arrojado una proporción notablemente mayor de tweets con connotación negativa en comparación con los positivos. Esta disparidad plantea la pregunta de si refleja genuinamente el tono predominante de la plataforma o si se debe a la elección de una sección particularmente densa en datos negativos. Explorar las razones detrás de esta inclinación hacia la negatividad resulta crucial para comprender la dinámica de uso de la plataforma y proporciona una perspectiva esencial para la interpretación precisa de los resultados del análisis de sentimientos.

## 5. Conclusión

En resumen, el uso de redes neuronales convolucionales para el análisis de sentimientos en textos mediante Python ha demostrado ser efectivo. Las CNN, originalmente diseñadas para tareas de visión por computadora, han sido adaptadas exitosamente al procesamiento de texto, aprovechando su capacidad para capturar patrones locales y características relevantes en datos secuenciales.

La implementación de CNN para análisis de sentimientos en Python, a menudo con bibliotecas como TensorFlow o PyTorch, ha facilitado el desarrollo y la experimentación con modelos. Estas redes pueden aprender representaciones jerárquicas de características, lo que las hace capaces de identificar patrones complejos en el lenguaje natural.

Sin embargo, al igual que con otros enfoques, la calidad del conjunto de datos y el preprocesamiento del texto son factores cruciales. Además, aunque las CNN pueden ser eficaces para capturar características locales, pueden enfrentar desafíos en la captura de dependencias a largo plazo en el texto.

En conclusión, el uso de redes neuronales convolucionales para el análisis de sentimientos en Python es una opción válida y eficaz, con sus propias ventajas y desafíos. La elección entre RNN y CNN dependerá de la naturaleza específica del problema y de las características del conjunto de datos.